



TRABAJO FIN DE GRADO  
GRADO EN INGENIERIA INFORMATICA

# Título

---

Subtítulo

Francisco de Asís Carrasco Conde

---

**Autor**

Francisco de Asís Carrasco Conde

**Directora**

María José Rodríguez Fórtiz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, Junio de 2025

**Título**  
**Subtítulo**

Francisco de Asís Carrasco Conde

**Palabras clave:** *software libre*

**Resumen**



## Same, but in English

Student's name

**Keywords:** *open source, floss*

**Abstract**



---

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación **TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI **XXXXXXXXXX**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .





---

D. **María José Rodríguez Fórtiz**, Profesora del Departamento de Lenguajes y Sistemas Informáticos

**Informo:**

Que el presente trabajo, titulado ***Nombre de la App***, ha sido realizado bajo mi supervisión por **Francisco de Asís Carrasco Conde**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2025.

**El/la director(a)/es:**

**María José Rodríguez Fórtiz**



## Agradecimientos



# Índice general

<b>1. Introducción</b>	<b>19</b>
1.1. Justificación . . . . .	19
1.2. Objetivos . . . . .	20
1.3. Estructura de la Memoria . . . . .	21
<b>2. Estado del arte</b>	<b>23</b>
2.1. Dominio del problema . . . . .	23
2.2. Aplicaciones similares . . . . .	24
2.3. Metodologías potenciales a aplicar . . . . .	28
2.4. Tecnologías potenciales para usar . . . . .	29
2.4.1. Base de datos . . . . .	29
2.4.2. Backend . . . . .	31
2.4.3. Frontend . . . . .	33
<b>3. Propuesta</b>	<b>35</b>
3.1. Descripción detallada . . . . .	35
3.2. Elección de tecnologías . . . . .	37
3.3. Backend . . . . .	37
3.4. Frontend . . . . .	37
3.5. Base de datos . . . . .	37
3.6. Diagrama de arquitectura . . . . .	38
3.7. Metodología utilizada . . . . .	38
3.8. Temporización . . . . .	38
3.9. Seguimiento del desarrollo . . . . .	39
3.9.1. Iteración 0 . . . . .	39

3.9.2.	Iteración 1 . . . . .	82
3.9.3.	Iteración 2 . . . . .	84
3.9.4.	Iteración 3 . . . . .	90
3.9.5.	Iteración 4 . . . . .	95
3.9.6.	Iteración 5 . . . . .	103

## Índice de figuras

2.1. Fitbit . . . . .	25
2.2. Apple fitness . . . . .	26
2.3. Strava . . . . .	27
2.4. Pulsera whoop . . . . .	28
2.5. Tabla comparativa tipos de bases de datos . . . . .	31
2.6. Tabla comparativa frameworks para el backend . . . . .	32
2.7. Tabla comparativa frameworks para el frontend . . . . .	34
3.1. Tabla comparativa . . . . .	36
3.2. Tabla comparativa . . . . .	38
3.3. Pagina inicial . . . . .	42
3.4. Inicio de sesion . . . . .	43
3.5. Crear cuenta 1 . . . . .	44
3.6. Crear cuenta 2 . . . . .	45
3.7. Menu principal . . . . .	46
3.8. Lista ejercicios . . . . .	47
3.9. Lista ejercicios filtrada . . . . .	48
3.10. Datos ejercicio . . . . .	49
3.11. Pop up graficar segun tiempo . . . . .	50
3.12. Pop up establecer meta . . . . .	50
3.13. Pop up crear ejercicio . . . . .	51
3.14. Lista rutinas . . . . .	52
3.15. Lista rutinas filtradas . . . . .	53
3.16. Datos rutina modificable . . . . .	54
3.17. Datos rutina no modificable . . . . .	54
3.18. Modificar ejericios rutina . . . . .	55
3.19. Cambiar orden ejercicios rutina . . . . .	56
3.20. Ejercicios rutina descargada . . . . .	57

3.21. Pop up crear rutinas . . . . .	58
3.22. Opciones de perfil usuario . . . . .	59
3.23. Pop up de confirmacion . . . . .	60
3.24. Pop up de confirmacion resumir datos . . . . .	60
3.25. Buscar usuario . . . . .	61
3.26. Buscar usuario filtrado . . . . .	62
3.27. Rutinas de un usuario . . . . .	63
3.28. Widget descargar rutina . . . . .	64
3.29. Widget rutina descargada . . . . .	64
3.30. Buscar rutinas por su nombre filtrada . . . . .	65
3.31. Entrenamiento de un día determinado . . . . .	66
3.32. Descanso en un día determinado . . . . .	67
3.33. Frame 29.png . . . . .	68
3.34. Lista ejercicios del entrenamiento actual . . . . .	69
3.35. Primera serie flexiones . . . . .	70
3.36. Realizando flexiones . . . . .	71
3.37. Fin de la serie(Descanso no completado) . . . . .	72
3.38. Fin de la serie(Descanso completado) . . . . .	73
3.39. Acabar ejercicio o añadir serie . . . . .	74
3.40. Realizando ejercicio midiendo tiempo . . . . .	75
3.41. Fin entrenamiento . . . . .	76
3.42. Metas cumplidas . . . . .	77
3.43. Datos entrenamiento terminado . . . . .	78
3.44. Chat con IA . . . . .	79
3.45. Datos detallados entrenamiento terminado . . . . .	80
3.46. Pop up datos de un ejercicio en entrenamiento . . . . .	81
3.47. BD local iteracion 1 . . . . .	84
3.48. Pantalla nueva . . . . .	85
3.49. Pantalla antigua . . . . .	85
3.50. Comparación entre la pantalla nueva y la anterior . . . . .	85
3.51. BD local iteracion 2 . . . . .	89
3.52. BD backend iteracion 2 . . . . .	90
3.53. BD backend iteracion 3 . . . . .	94
3.54. Pantalla Calendario . . . . .	98
3.55. Flujo entrenamiento . . . . .	101
3.56. BD local iteración 4 . . . . .	102



## Índice de tablas



# Capítulo 1

## Introducción

### 1.1. Justificación

En este TFG se va a desarrollar una aplicación que ayude a las personas a realizar ejercicio físico de forma controlada y a monitorizar las metas que se van alcanzando. Primero de todo, ¿qué lleva a hacer este TFG? Para empezar, es verdad que hay muchas apps en los repositorios que están pensadas para ser usadas por usuarios de gimnasio, para hacer más fácil el seguimiento y evolución. Además la gente que suele hacer deporte de manera más informal o por hobby no suele tener los conocimientos o dispositivos para hacer mediciones de calidad e interpretarlo correctamente, por lo que necesitarían una aplicación más sencilla tipo agenda para planificar y monitorizar sus ejercicios.

Otra problemática que encontramos en las apps existentes suele ser la falta de accesibilidad, pensando en usuarios con algún tipo de discapacidad que quieran realizar ejercicios. Por ejemplo, encontramos que los scrolls abundan, hay eventos no controlados por el usuario, colores confusos para daltónicos, a veces hay ausencias de iconos asociados a botones y listas, y un amplio etcétera.

Otras carencias que se observan es que no suelen incluir la funcionalidad de ver entrenamientos recomendados por otros. La gente suele buscarlos en redes sociales o videos que se encuentran en la red, y muchas veces en estos videos se ponen a dar rodeos para rascar más tiempo, así es como el usuario pierde tiempo para que a lo mejor no sea el entrenamiento que el andaba buscando. Lo ideal sería que los ejercicios estuvieran bien organizados en rutinas y que estas pudieran compartirse en la aplicación.

Me veo en la necesidad de hacer esta app para dotar de una herramienta simplificada, fácil de manejar, que permita compartir información entre usuarios

de forma eficaz y accesible. Simplificada porque la información que manejo se trata de una forma fácil e intuitiva, con un poco de experiencia en el deporte se sabe qué significa cada parámetro a medir en un ejercicio. Fácil de manejar, dado que el usuario solo tendrá que, por ejemplo, anotar el número de veces que levanta una pesa y anotarlo en la app. Compartir información eficazmente, porque en una sección de la app habrá una parte formato red social, que permitirá tanto buscar usuarios que comparten entrenamientos, como los propios entrenamientos en sí, acompañados de su descripción en la que el creador da una breve información acerca del entrenamiento. Accesible, porque se seguirán guías de diseño para facilitar el uso por usuarios de diversas capacidades.

Una vez explicado esto, ¿cómo se le dará soporte a los usuarios? Creando una app que le ayude de la siguiente forma: Ayudándole a planificar o usar rutinas de entrenamientos compuestas por series de ejercicios; Facilitándole la monitorización y supervisión de su realización, al poder introducir metas a alcanzar en parámetros, que pueden ser repeticiones/peso/distancia/tiempo, y que se pueden medir de forma independiente o al mismo tiempo. También guardando la cantidad de series de un ejercicio realizadas en un entrenamiento con sus respectivos parámetros, y contabilizándole al usuario el tiempo descansado, para facilitar su correcto entrenamiento. Además, si el usuario lo solicita se le enseñará un resumen de su rendimiento en cada ejercicio con respecto a la fecha en la que se realizó el mismo, para que vea su evolución respecto al tiempo.

También la app ayudará a hacer un control de las metas que se proponga el usuario, es decir, si el usuario se propone bajar de peso o aumentar su rendimiento en x ejercicio, la app le recordará las metas que se autopropuso y le avisará cuando las cumpla.

## 1.2. Objetivos

Quiero que la app sea un historial interactivo del deporte realizado por el usuario.

Los objetivos de este trabajo de fin de grado son:

- Analizar algunas apps del mercado, así como sus características, qué ofrecen, su costo para el usuario y las valoraciones de los usuarios finales, para aclarar qué es lo que buscan los usuarios en este tipo de apps y considerar esas necesidades en el software a desarrollar.
- Investigar sobre qué herramientas usar para la implementación de la base de datos, backend, frontend, y técnicas y metodologías para dar un desarrollo de calidad al software y para ayudar a la sostenibilidad del sistema en el tiempo.

- Conocer y aprender a usar herramientas actuales y punteras para el desarrollo de aplicaciones móviles.
- Desarrollar una aplicación multiplataforma que cubra todos los requisitos necesarios para dar soporte a la planificación y realización de entrenamientos y monitorización de rutinas de ejercicio físico.
- Tratar de obtener un software lo más accesible posible.

### 1.3. Estructura de la Memoria

El 1º capítulo(**Introducción**) es una descripción sobre lo que se va a abordar y qué ideas iniciales se tienen acerca del software que se va a desarrollar y algunos aspectos importantes del gremio en el que se usaría.

El 2º capítulo(**Estado del arte**), es un análisis de cómo está el ámbito sobre el que se va a desarrollar este trabajo, es decir, apps del mercado(sus prestaciones y funciones más importantes) y frameworks que se podrían emplear para el desarrollo, así como sus ventajas y desventajas.

El 3º capítulo,

El 4º capítulo(**Conclusiones y trabajos futuros**),



## Capítulo 2

### Estado del arte

#### 2.1. Dominio del problema

Para comenzar, es necesario hablar un poco sobre conceptos generales en el mundo del deporte, definiendo algunos conceptos. Un ejercicio es la repetición de un movimiento varias veces para estimular uno o varios músculos. Las rutinas, entendámoslos como la colección de distintos ejercicios destinados a entrenar una parte o varias del cuerpo, a las rutinas también se les llama entrenamientos, pero en la app se referirá a rutinas como la anterior definición y los entrenamientos serán el hecho de haber realizado una rutina. Ejemplo, el entrenamiento del día 6 es hacer la rutina de flexiones. Una serie es cuando dentro de un ejercicio repetimos ese movimiento un número determinado de veces. Entre series se hacen descansos. Una repetición, es la realización del movimiento de ese ejercicio en una serie, es decir, si por ejemplo estoy haciendo flexiones, hago 12, descanso, hago 10, descanso, hago 9 y ya no hago más flexiones, dentro de mi entrenamiento se vería así:

##### Rutina de entrenamiento 1:

- Ejercicio 1
  - Serie 1: X repeticiones
  - Descanso
  - Serie 2: X repeticiones
- Flexiones
  - Serie 1: 12 repeticiones

- Descanso
  - Serie 2: 10 repeticiones
  - Descanso
  - Serie 3: 9 repeticiones
- Ejercicio 3
- Serie 1: X repeticiones
  - Descanso
  - Serie 2: X repeticiones

Las metas son objetivos que se pone el usuario en un ejercicio, por ejemplo, si un usuario desea llegar a hacer 10 flexiones mínimo en una serie la proxima vez que entrene, es que se ha puesto una meta de 10 flexiones.

El compartir una rutina es publicar los detalles de una rutina para que otra persona la pueda hacer en sus entrenamientos.

En este gremio del deporte es muy importante la planificación, ya que si se entrena muy de seguido un músculo te puedes llegar a lesionar o hacer que los entrenamientos sean inútiles, lo que se conoce como sobrentrenamiento. Por su contraparte, es importante no dejar de entrenar todas las zonas del cuerpo, ya que te puede lastrar una musculatura debil para otros tipos de entrenamientos. También es importante planificar los días de descansos, para evitar entrenamientos con el cuerpo fatigado.

Ya dentro de los entrenamientos también es importante respetar los descansos. Dado que si se descansa menos tiempo del debido puede provocar lesiones de muchos tipos, incluido neuronales dado al estrés que sufre el cerebro durante el deporte.

Aunque hagamos correctamente lo mencionado en los 2 párrafos anteriores, puede que el deportista no vea los frutos de sus entrenamientos porque o bien no se acuerda de su rendimiento de hace 1 mes y no sabe si es el mismo o no, o simplemente es un sentimiento placebo. Por ello sería util tener un historial de entrenamiento de fácil acceso o incluso gráficas, para ver si el deportista a mejorado su rendimiento o no.

## 2.2. Aplicaciones similares

En esta sección describiremos algunas de las aplicaciones que existen similares a la propuesta. Las que permiten mediciones de constantes relacionadas durante el entrenamiento físico suelen ser de pago.

Ejemplos de aplicaciones con precios mensuales:



- **Fitbit Premium: €9.99**, esta app ofrece una interfaz para medir constantes usando un smartwatch con sensor cardíaco (ritmo cardíaco, calorías quemadas, pasos, ect). En lo que respecta que es el entrenamiento, solo ofrece una serie de entrenamiento prefijados, además no permite guardar resultados.

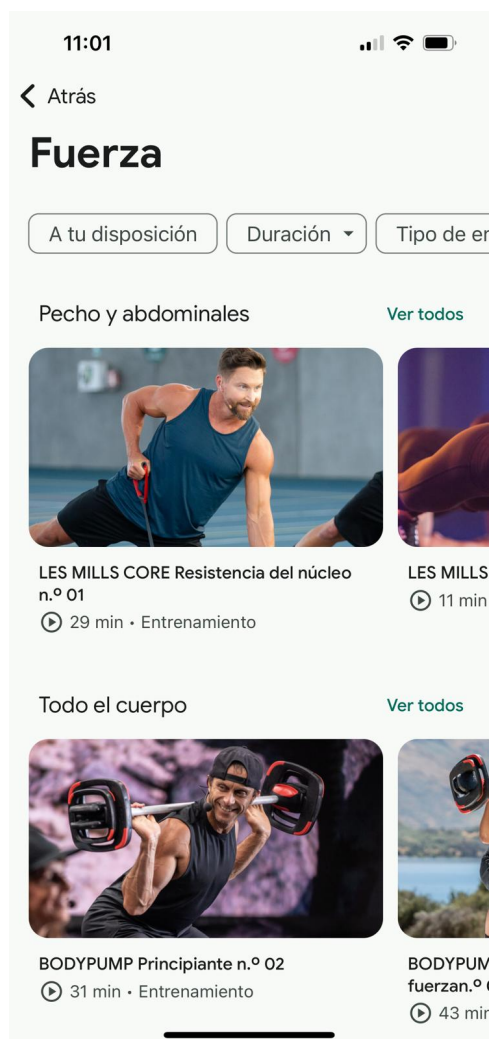


Figura 2.1: Fitbit

- **Apple Fitness+: €9.99**, en su versión gratuita nos permite guardar todas las variables permitidas con un smartwatch, las registra y va aconsejando, si detecta que tienes estrés salta un aviso, poco movimiento, pulsaciones anormales, etc. En la versión de pago añade los entrenamientos, pero vuelve a la misma problemática, son entrenamientos prefijados y no permite

guardar tu rendimiento.



Figura 2.2: Apple fitness

- **Strava Premium: €5.99**, es una app más enfocada a running, pero permite medir variables relacionadas con este tipo de entrenamientos, kilómetros recorridos, ritmo, ruta recorrida, tiempo, etc. Es muy completa solo para ese tipo de entrenamientos.

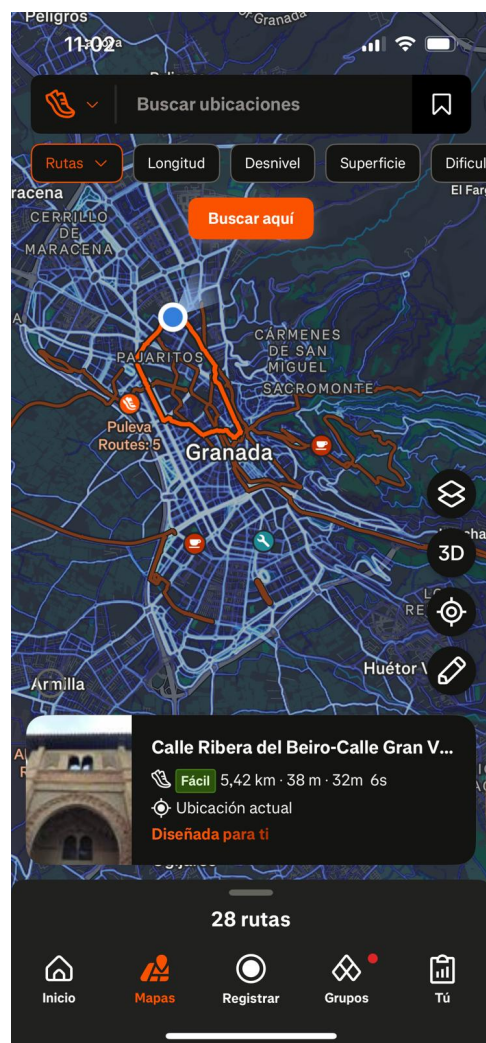


Figura 2.3: Strava

- **Whoop: €28**, de las aplicaciones esta es de lejos la más completa, permite medir, estrés, cansancio del usuario, calidad del entrenamiento, calidad de sueño, recuperación, edad de tu cuerpo según tus constantes, IA especializada para chatear y recibir consejos, etc. Sin embargo, es la que tiene la mensualidad más cara y solo funciona con un reloj de su marca, es decir, el usuario está obligado a comprar ese reloj si quiere usar dicha app, el reloj en concreto vale 200€ mínimo.



Figura 2.4: Pulsera whoop

Es verdad que algunas tienen versión gratuita, pero no ofrecen la totalidad de sus funcionalidades, de hecho, estas versiones suelen ser extremadamente limitadas. Otras aplicaciones como Garmin Connect, sí que son gratuitas, pero te obligan a usar un smartwatch de la marca Garmin, cuyo precio no baja de los 300€.

## 2.3. Metodologías potenciales a aplicar

Existen muchas formas de desarrollar software con sus respectivas ventajas, desventajas y forma de trabajar. En este caso, no voy a investigar sobre metodologías que usen la idea de prototipos (software que puede ser probado por el cliente durante el desarrollo):

- **Cascada:** Es una forma de desarrollar de las más clásicas y lineales. Se hace una secuencia de fases de desarrollo, las cuales suelen ser diseño requisitos, diseño, implementación, pruebas, implementación y mantenimiento. Es una metodología fácil de entender y aplicar, eficaz para plazos de entrega estrictos. Sin embargo, no es flexible, si se encuentran fallos en la planificación o alguna de las fases, no hay mucho tiempo de reacción.
- **Desarrollo Ágil:** En lugar de una secuencia rígida de fases, se promueve la flexibilidad, ya que se combina con reuniones con el cliente para obtener feedback del software funcional desarrollado entregado en dichas reuniones (no es un prototipo ya que solo es una demostración, no se le entrega para su uso durante el desarrollo), permitiendo corregir errores de planificación o en el propio software durante el desarrollo, sin perder mucho tiempo. Es necesario un buen feedback del cliente.
- **Lean Software Development:** Se basa en la idea de descartar todo aquello que no aporte valor para el cliente y quitarle importancia al de-

sarrollo para ganar calidad y sostenibilidad en el tiempo al software. Esto se refleja en que primero se hace una investigación a fondo de todas las herramientas posibles a usar, su consecuente aprendizaje y en lo más tardío de todo esta la elección de que herramientas usar. Una vez hecha la elección, ya se puede empezar con el desarrollo.

- **V Model:** Es como el método cascada, pero antes de pasar a la siguiente fase se realizan las pruebas de la actual y no se procede hasta que estas estén válidas. Sigue teniendo las mismas desventajas que el modelo de cascada, pero es ideal para proyectos en los que es necesario una alta calidad.
- **Modelo en espiral:** En cada fase se realiza planificación, diseño, desarrollo y evaluación de riesgos. Es muy flexible y esta analizando constantemente los riesgos, pero se requiere cierta experiencia para su correcto empleo.

## 2.4. Tecnologías potenciales para usar

Para hablar de las tecnologías a emplear, lo voy a separar en las 3 partes importantes en un desarrollo como el que se va a abordar: la base de datos, el backend y el frontend.

### 2.4.1. Base de datos

En este punto vamos a hablar de las ventajas y desventajas entre usar BDs relacionales y no relacionales, así como las herramientas a emplear en cada una para desarrollar estos servicios.

**Relacionales:** Las bases de datos relacionales o SQL, como bien sabemos nos permiten guardar datos de una manera bien definida y estructurada, permitiendo asegurar la integridad de la información almacenada. Son muy buena opción, si se prefiere una escalabilidad vertical, es decir, aumentar la capacidad de procesamiento del servidor que va a albergar la BD. Algunas de los SGBDS existentes en la actualidad son:

- MySQL
- MariaDB
- Oracle Database
- SQL server(Microsoft)

- PostgreSQL

**No relacionales:** También llamadas NoSQL, sirven para trabajar con estructuras de datos semidefinidas o no definidas. Esto quiere decir que no siguen un esquema rígido, lo cuál puede complicar consultas complejas, pero permiten una gran escalabilidad horizontal, permiten añadir más servidores para que operen con la misma base de datos, por tanto aumentar el número de peticiones que puede atender. Cabe decir que dentro de este tipo de bases de datos existen varios subtipos, cada uno especializado en una cualidad:

- Documentales(p.e. MongoDB): guardan los datos en un JSON
- Columnares(p.e. Cassandra): los datos no se guardan en filas, sino en columnas ,ideal para tratar grandes volúmenes de datos por columnas
- Clave-Valor(p.e. DynamoDB): acceso rápido a los datos
- Graficas(p.e. Amazon Neptune): para tratar relaciones complejas

	SQL	NoSQL
<b>Estructura de datos</b>	Datos estructurados con esquema rígido (tablas, filas, columnas)	Datos semi-estructurados o no estructurados (JSON, columnas, clave-valor)
<b>Modelo de datos</b>	Relacional	Varios modelos: documental, columnares, clave-valor, gráficas
<b>Integridad de datos</b>	Alta	Variable
<b>Escalabilidad</b>	Vertical (mejorar hardware de un solo servidor)	Horizontal (añadir más servidores para distribuir carga)
<b>Consultas complejas</b>	Soportadas y optimizadas	Menos eficientes o más complicadas para consultas complejas
<b>Casos de uso típicos</b>	Sistemas con datos estructurados, transacciones, bancos, ERP	Big data, datos no estructurados, aplicaciones web, IoT, análisis en tiempo real
<b>Flexibilidad en esquema</b>	Baja	Alta
<b>Rendimiento en volumen alto</b>	Puede verse limitado por la escalabilidad vertical	Muy buena para volúmenes grandes y distribuidos

Figura 2.5: Tabla comparativa tipos de bases de datos

### 2.4.2. Backend

En el desarrollo del backend existen varios frameworks que nos van a permitir un desarrollo rápido, eficaz y de calidad. Nuestro backend se va encargar principalmente de recibir y realizar peticiones de los clientes y/o consultas a la BD. Para ello se han investigado las siguientes herramientas:

- Express.js(p.e. Node.js): muy escalable y alto rendimiento

- Django(p.e. Python): ideal para sistemas a gran escala
- Ruby on rails: el mejor en velocidad de desarrollo

Existen más frameworks, pero he decidido investigar solo sobre aquellos que me permitan un desarrollo más ágil, el resto de herramientas como Spring Boot(Java) o Laravel(PHP), también son buenas herramientas, pero tengo más familiaridad con algunas de las herramientas anteriores.

	<b>Express.js (Node.js)</b>	<b>Django (Python)</b>	<b>Ruby on Rails</b>
<b>Lenguaje base</b>	JavaScript	Python	Ruby
<b>Rendimiento</b>	Alto, muy escalable	Muy bueno, adecuado para gran escala	Bueno, optimizado para desarrollo rápido
<b>Velocidad de desarrollo</b>	Rápida, modular y flexible	Rápida, con muchas funcionalidades integradas	Muy rápida, con convenciones que agilizan el desarrollo
<b>Facilidad de aprendizaje</b>	Fácil para desarrolladores JS	Moderada, requiere conocer Python	Moderada, requiere conocer Ruby
<b>Comunidad y soporte</b>	Amplia comunidad, mucha documentación	Comunidad sólida y madura	Comunidad activa y enfocada en productividad
<b>Casos de uso típicos</b>	APIs REST, aplicaciones en tiempo real, microservicios	Aplicaciones web completas, proyectos grandes	Proyectos con desarrollo ágil, startups, MVPs
<b>Flexibilidad</b>	Muy alta, minimalista	Completo y con muchas herramientas incluidas	Alta, con convenciones fuertes para facilitar buenas prácticas
<b>Herramientas integradas</b>	Básico, depende de módulos externos	Incluye ORM, sistema de autenticación, administración	Incluye ORM, sistema de rutas, validaciones integradas

Figura 2.6: Tabla comparativa frameworks para el backend



### 2.4.3. Frontend

Como se dijo en la introducción, uno de los objetivos es que el software sea multiplataforma, así que se investigará sobre frameworks que me permitan esa capacidad:

- Flutter(lenguaje Dart): de las manos de Google, esta herramienta permite un desarrollo rápido ya que nos permite el hot reload y no tener que recompilar la app cada vez que haya un cambio. También es muy personalizable en lo que respecta a la UI.
- React Native(lenguaje JavaScript): es un framework que extiende React, permitiendo hacer sentir aplicaciones en js como si fueran nativas.
- Xamarín(Lenguajes C#): de Microsoft, esta herramienta es ideal para desarrolladores familiarizados con C# y .NET, también es idóneo para el alto rendimiento y acceso a HW nativo.

	<b>Flutter (Dart)</b>	<b>React Native (JavaScript)</b>	<b>Xamarin (C#)</b>
<b>Lenguaje base</b>	Dart	JavaScript (extensión de React)	C# (.NET)
<b>Velocidad de desarrollo</b>	Alta, con hot reload para ver cambios al instante	Alta, con hot reload	Moderada, requiere compilación
<b>Experiencia UI</b>	Muy personalizable, widgets propios para UI nativa	UI basada en componentes nativos	Acceso a controles nativos para UI
<b>Rendimiento</b>	Muy cercano a nativo	Bueno, aunque puede depender de puentes JS nativos	Muy cercano a nativo, alto rendimiento
<b>Facilidad de aprendizaje</b>	Requiere conocer Dart	Fácil si se conoce JavaScript y React	Más fácil para desarrolladores C#.NET
<b>Acceso a hardware nativo</b>	Sí, mediante plugins	Sí, mediante módulos nativos	Sí, acceso directo al hardware nativo
<b>Comunidad y soporte</b>	Creciente, fuerte respaldo de Google	Amplia comunidad y ecosistema React	Comunidad sólida dentro del ecosistema Microsoft
<b>Casos de uso típicos</b>	Aplicaciones móviles con UI personalizada y multiplataforma	Apps móviles multiplataforma con rapidez y flexibilidad	Apps multiplataforma con integración profunda en Windows y otras plataformas

Figura 2.7: Tabla comparativa frameworks para el frontend

## Capítulo 3

### Propuesta

#### 3.1. Descripción detallada

La app presentada daría la mayoría de funcionalidades de pago de una manera gratuita y aporta su funcionalidad de medir de forma personalizada el rendimiento del deportista. También se añade una IA, una funcionalidad que no vista en muchas aplicaciones del sector y si existen, son de pago. La función de la inteligencia artificial sería la de aconsejar al usuario cuando este lo necesite, sobre su rendimiento y/o parámetros medidos durante su entrenamiento. Para clarificar las diferencias con el resto de apps del mercado:

X	Registra datos alimenticios	Mide rendimiento de un entrenamie	Puedes hacer tus propios entrenamie	Puedes compartir entrenamientos/conten	Es necesario usar un smartwatch	Tienes una IA consultiva	Requiere pagar una subscripción	Permite controlar el peso del usuario	Puede ser usada offline	Establece metas para tus entrenamie
Strava	No	Solo de running	Solo de running	Solo running	No	No	Existe una versión gratuita limitada	No	No	Solo en los de runniing
Apple Fitness	Si	No	No	No	No	No	Existe una versión gratuita limitada	Si	Solo la parte gratuita	No
Fitbit	SI	No	No	No	No	No	No	Si	Solo la parte gratuita	No
Whoop	No	Si	No	No	Si	Si	Si	Si	Si	No
App presentada	No	Si	Si	Si	No	Si	No	Si	Casi en su plenitud	Si

Figura 3.1: Tabla comparativa

Francisco de Asís Carrasco Conde

## 3.2. Elección de tecnologías

## 3.3. Backend

Se usará Express.js(Node.js), la principal razón que se tiene para optar por esta tecnología, es la gran familiaridad que se tiene con el entorno de JavaScript, se conocen las tecnologías y la forma de implementación, su gran flexibilidad para implementar muchas funcionalidades diversas también fué una buena baza para escoger esta herramienta. Django fué la primera descartada, porque como bien dice en la tabla de la sección anterior(fig. 2.6) está más enfocado a proyectos de gran envergadura. Ruby on Rails fue la que más dudas planteó, como se dijo anteriormente es muy buena opción para las metodologías ágiles y como se dirá más adelante, se optará por una metodología ágil para el desarrollo. Pero al final se descartó por el desconocimiento de la herramienta.

## 3.4. Frontend

Se ha decidido usar Flutter, dado que Xamarin es buena herramienta, pero está más enfocada a un desarrollo en un ambiente Microsoft. React Native también es una buena herramienta, fue principalmente la que más dudas sembró, ya que como se dijo previamente se usará un backend basado en Express.js, esto entonces permitiría unificar entornos lo cual favorecería al desarrollo. pero se descartó por lo personalizable que son las IUs desarrolladas en Flutter respecto a esta tecnología.

## 3.5. Base de datos

Para las bases de datos, tanto para el backend como para la local de la app se usará una SQL, dado a la familiaridad que se dispone con esta tecnología y la necesidad de unificar el tipo de tecnología del servidor y de la app. Esto anterior se debe a que como bien se ha dicho se usará Flutter como entorno frontend y la herramienta que se facilita para el almacenamiento local en este es SQLite, basada en un modelo relacional.

Los modelos NoSQL, se descartaron principalmente porque las relaciones entre elementos suelen ser más complejas y en este proyecto pueden enrevesarse fácilmente.

### 3.6. Diagrama de arquitectura

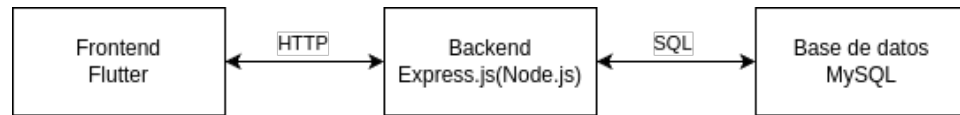


Figura 3.2: Tabla comparativa

### 3.7. Metodología utilizada

Para el desarrollo de esta app, se usará una metodología *ágil* tipo **SCRUM**. Se decidió usar esta porque permite corregir fallos en la velocidad de diseño y/o planificación de forma eficiente y sin dañar el producto final. Esta metodología permite también entregar desarrollos funcionales a la tutora e ir validando las funcionalidades poco a poco mediante reuniones.

Las reuniones con la tutora serán las *sprint reviews*.

### 3.8. Temporización

La temporización se realizó el día 25 de marzo de 2025. La entrega del producto (este TFG) está prevista para el 16 de junio de 2025, es decir, 83 días, o lo que es lo mismo, casi 12 semanas. Si un sprint dura 2 semanas, habrá 6 sprints hasta la entrega final.

La iteración 0 se dedicará al diseño de pantallas y al repaso de las funcionalidades de la app, para concretar las historias de usuario y sus prioridades. La idea inicial es dividir la app en varios módulos y centrar cada sprint en cada uno de ellos:

1. Diagramas de la app y ejercicios
2. Rutinas, usuarios y sesión
3. Datos que ingresa el usuario
4. Flujo de entrenamiento
5. Inteligencia Artificial (IA)
6. Smartwatch

## 3.9. Seguimiento del desarrollo

### 3.9.1. Iteración 0

En esta primera iteración me centré en realizar los diseños de la app, considerando su accesibilidad. También concreté el *product backlog*, compuesto por 41 historias de usuario. En las historias no menciono ningún actor puesto que solo existe el usuario que realiza los ejercicios y el es protagonista en todas ellas, no hay ningún moderador.

A pesar de trabajar con SCRUM se realizarán todos los diseños de interfaces de usuario en este primer sprint para concretar con la tutora los requisitos principales de la aplicación a desarrollar en este TFG. En nuestro caso los diseños de las interfaces se usarán como herramientas de captura y especificación de requisitos. En cada iteración se podrán revisar si hay cambios.

La lista inicial de historias de usuario es la siguiente, y algunas incluyen tareas secundarias:

**SCRUM-1** Registrar peso por día

**SCRUM-2** Establecer peso objetivo

**SCRUM-3** Insertar/Borrar/Modificar ejercicio de la lista de ejercicios

**SCRUM-4** Buscar rutina en la lista del usuario

**SCRUM-5** Sustituir un ejercicio por otro en la rutina

**SCRUM-6** Insertar/Borrar/Modificar rutina

**SCRUM-7** Poner una meta en cada ejercicio

**SCRUM-8** Hacer grafica de los datos de los ejercicios

**SCRUM-9** Revisar datos para ver si el descanso es necesario

**SCRUM-10** Enseñar datos de una rutina a descargar

**SCRUM-11** Compartir mi rutina

**SCRUM-12** Guardar las repeticiones y series de todos los ejercicios de un entrenamiento

**SCRUM-13** Valorar el entrenamiento en base a la marca actual y la meta del usuario

**SCRUM-14** Monitorizar pulso en tiempo real

- SCRUM-15** Medir pulso en reposo y compararlo con datos de ejercicios
- SCRUM-16** Avisar de anomalías en el pulso de forma suave
- SCRUM-17** Obtener calorías quemadas
- SCRUM-18** Comprobar el equilibrio nervioso del usuario
- SCRUM-19** Realizar el flujo del entrenamiento
- SCRUM-20** Conectar con la IA para iniciar diálogo
- SCRUM-21** Crear/Borrar usuario
- SCRUM-22** Resumir datos
- SCRUM-23** Iniciar/Cerrar sesión
- SCRUM-24** Medir SpO2
- SCRUM-25** Interpretar constantes
- SCRUM-26** Buscar ejercicios en la lista de ejercicios
- SCRUM-27** Hacer la IU de la lista de los ejercicios
- SCRUM-28** Hacer la IU de creación de ejercicio
- SCRUM-29** Pop up de confirmación
- SCRUM-30** Hacer IU de datos ejercicio
- SCRUM-31** Crear IU de la lista de las rutinas
- SCRUM-32** Crear IU de pantalla inicial
- SCRUM-33** Crear IU menú principal
- SCRUM-34** Crear IU datos rutinas
- SCRUM-35** IU lista ejercicios de rutina modificable
- SCRUM-36** Crear IU para crear rutina
- SCRUM-36** Detectar metas cumplidas despues del entrenamiento
- SCRUM-37** Crear IU del perfil del usuario
- SCRUM-38** Detectar metas cumplidas en los ejercicios despues del entrenamiento
- SCRUM-39** Buscar rutinas para descargar



**SCRUM-40** IU del pop up de resumir datos

**SCRUM-41** IU desplegable rutina

A continuación se muestran los diseños creados en esta iteración:

**Inicio de sesion y crear usuario**

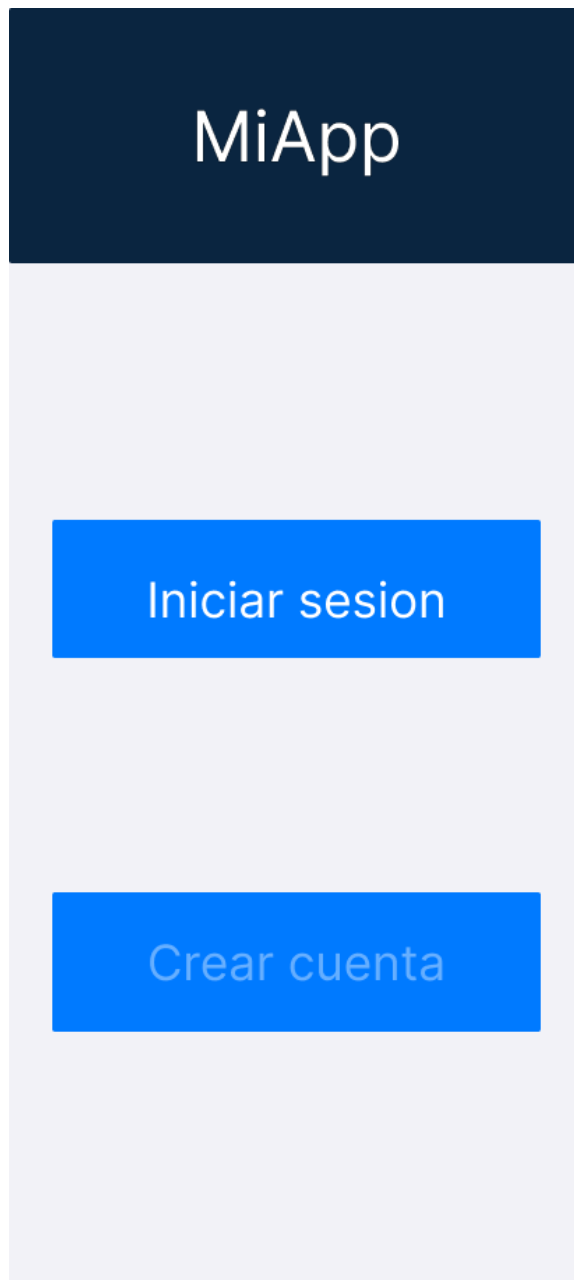
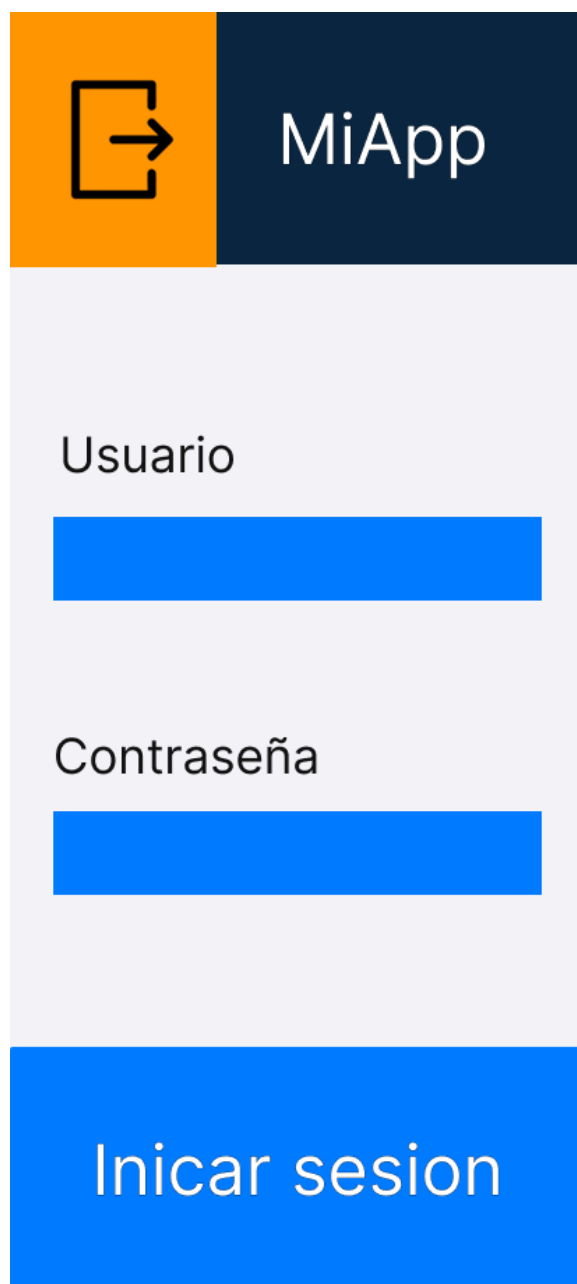
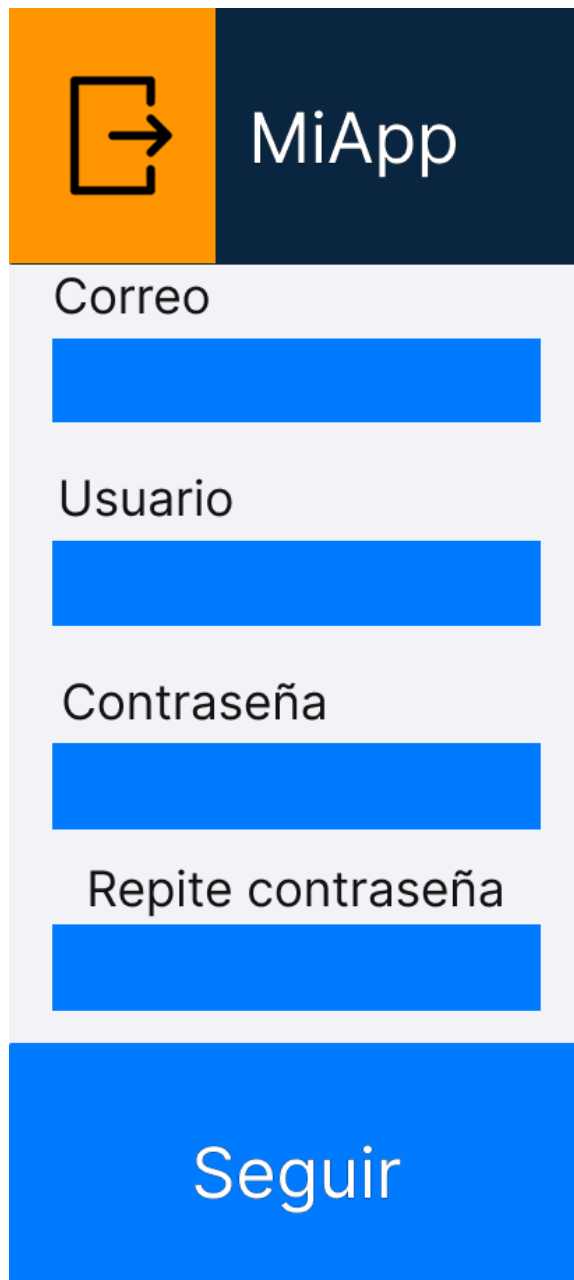


Figura 3.3: Pagina inicial



The image shows a mobile application login screen. At the top, there is a header bar with an orange square on the left containing a white icon of a document with an arrow pointing right, and a dark blue square on the right containing the text "MiApp" in white. Below the header is a light gray rectangular area containing two text input fields. The first field is labeled "Usuario" and the second is labeled "Contraseña". Both fields are represented by solid blue rectangles. At the bottom of the screen is a large blue rectangular button with the text "Inicar sesion" in white.

Figura 3.4: Inicio de sesion



The image shows a mobile application registration screen. At the top, there is a header with an orange square on the left containing a white icon of a document with an arrow pointing right, and a dark blue square on the right containing the text "MiApp" in white. Below the header is a light gray registration form. The form contains four text input fields, each preceded by a label: "Correo", "Usuario", "Contraseña", and "Repite contraseña". Each input field is represented by a solid blue rectangle. At the bottom of the form is a large blue button with the white text "Seguir".

Figura 3.5: Crear cuenta 1

**MiApp**

Fecha Nacimiento:

Género:

Peso:

Altura:

**Terminar**

Figura 3.6: Crear cuenta 2

Menu principal

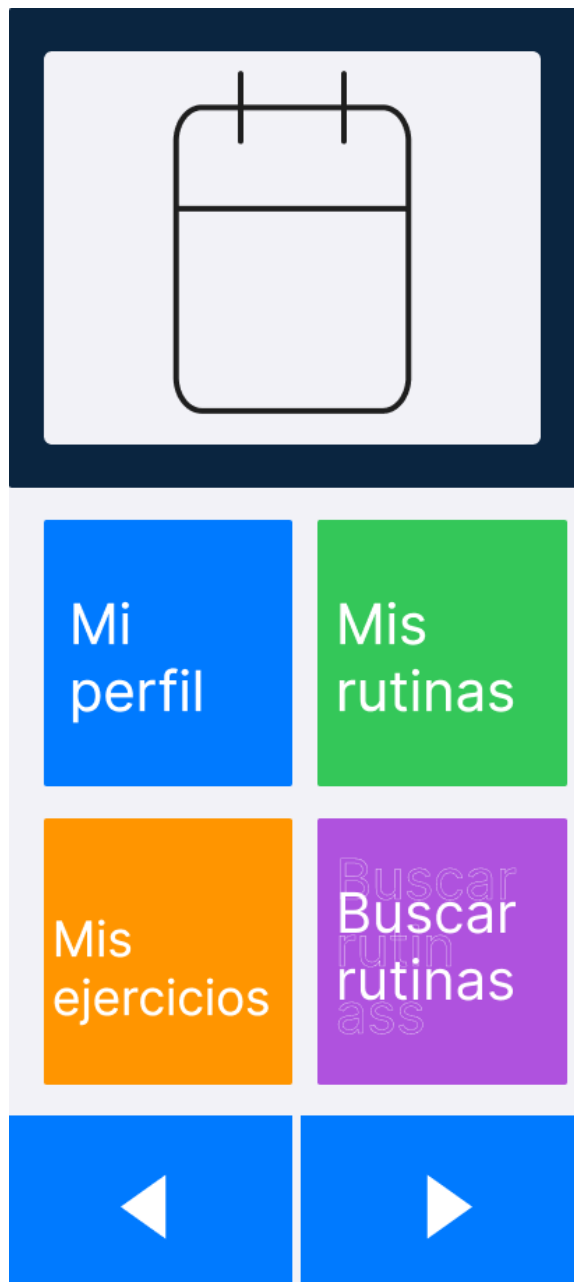


Figura 3.7: Menu principal

Lista ejercicios del usuario



Figura 3.8: Lista ejercicios

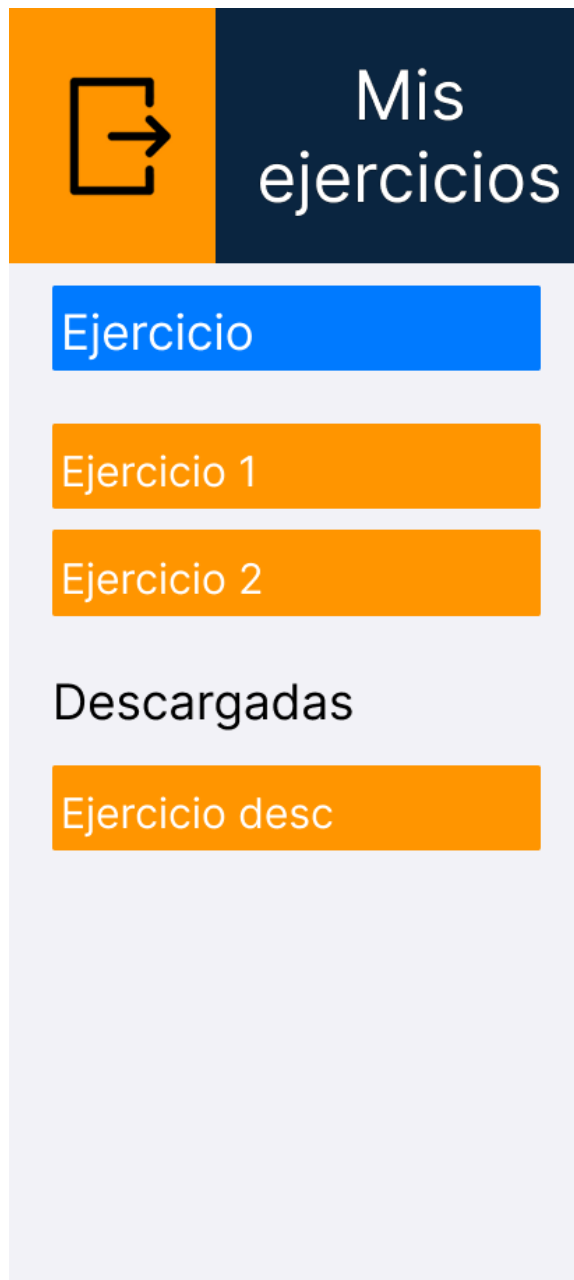




Figura 3.9: Lista ejercicios filtrada





Nombre  
ejercicio



Marca actual:  
7.4 repeticiones x 6kg

Valoración actual: 4.5 / 5

Graficar según  
tiempo

Meta actual:  
8 repeticiones x 6kg

Sin cumplir

Establecer  
meta

Figura 3.10: Datos ejercicio



Figura 3.11: Pop up graficar según tiempo

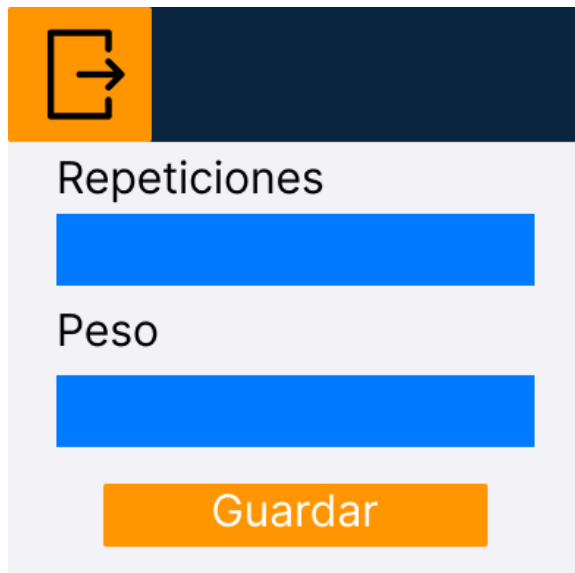
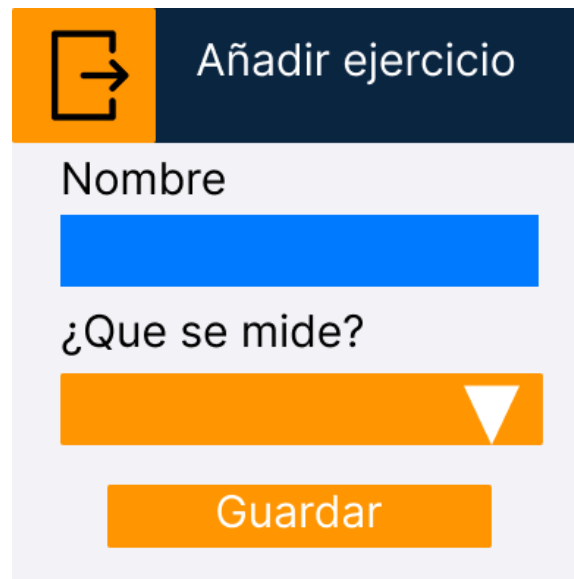
A pop-up window with a dark blue header bar containing a white icon of a square with an arrow pointing right. The main content area is light gray and contains the text "Repeticiones" above a blue horizontal bar, the text "Peso" above another blue horizontal bar, and an orange button labeled "Guardar" at the bottom.

Figura 3.12: Pop up establecer meta



Añadir ejercicio

Nombre

¿Que se mide?

Guardar

Figura 3.13: Pop up crear ejercicio

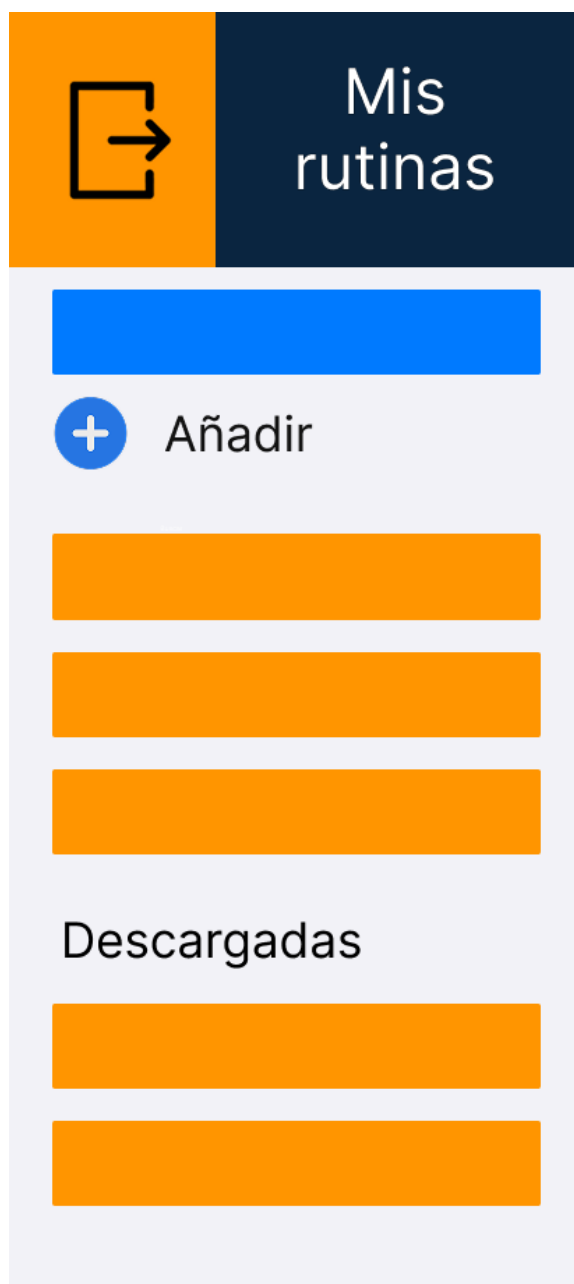


Figura 3.14: Lista rutinas

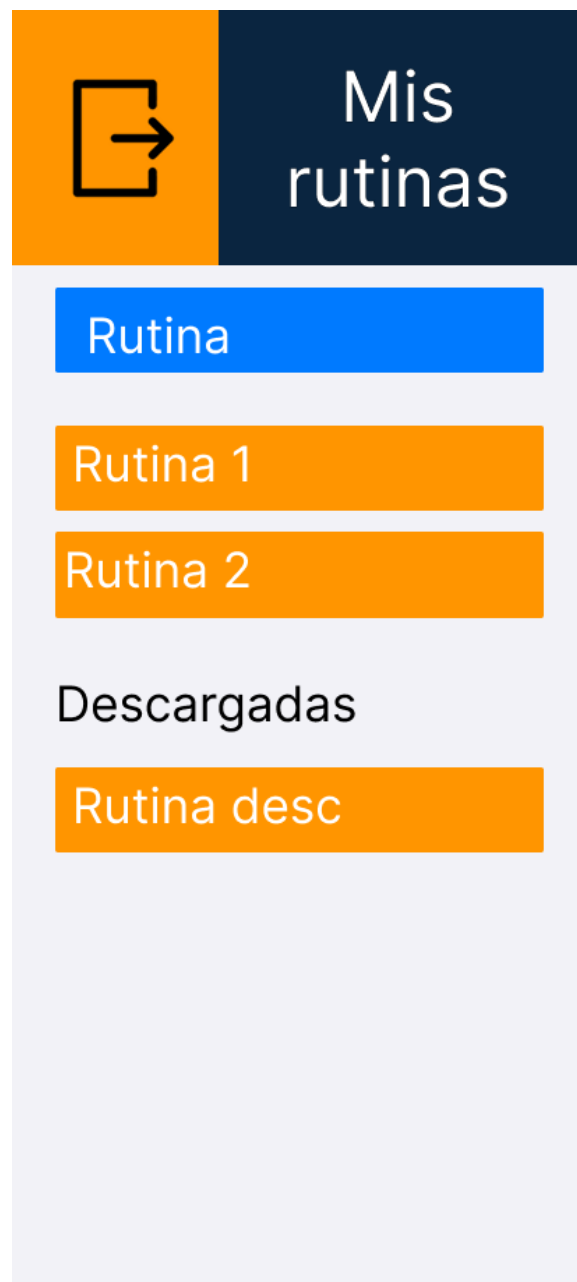


Figura 3.15: Lista rutinas filtradas



Figura 3.16: Datos rutina modificable

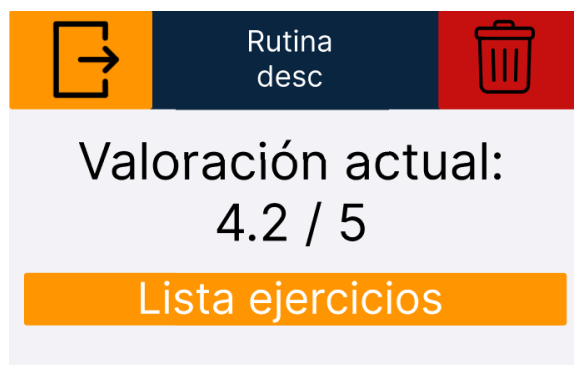


Figura 3.17: Datos rutina no modificable



Figura 3.18: Modificar ejercicios rutina

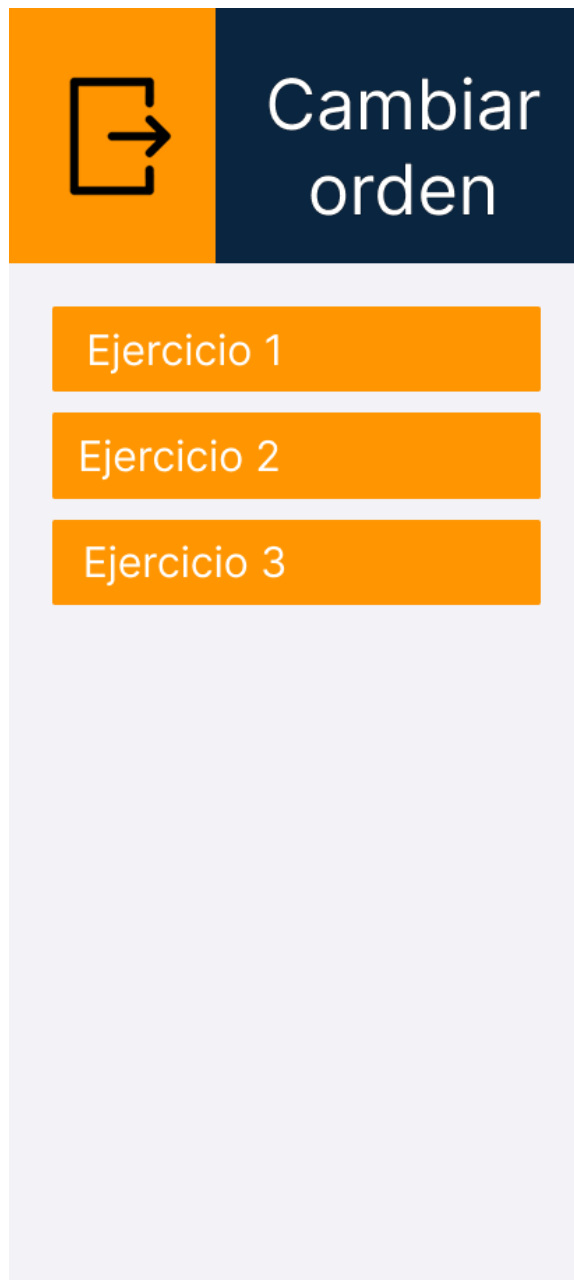


Figura 3.19: Cambiar orden ejercicios rutina



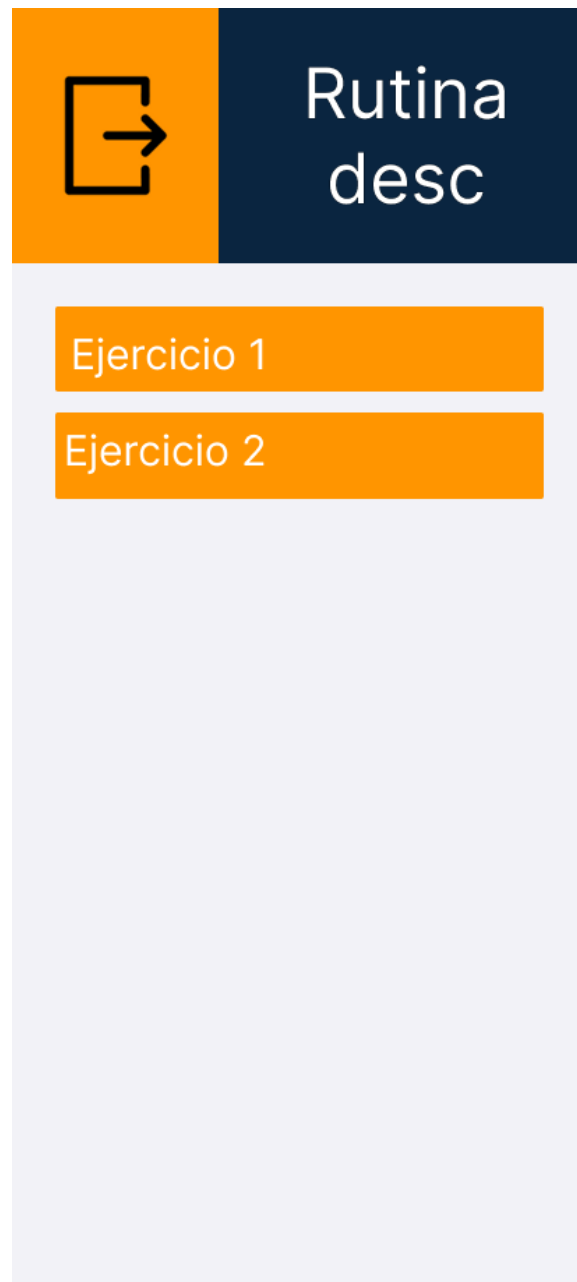
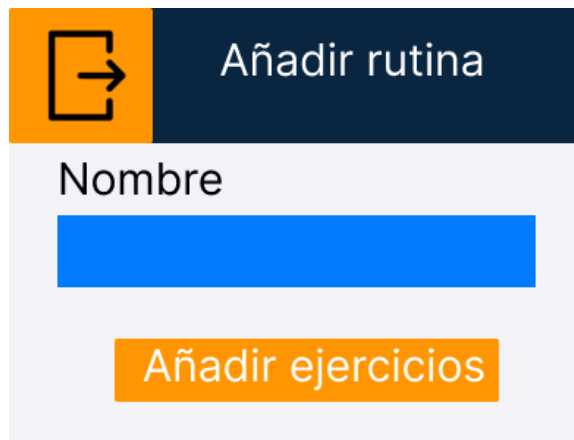


Figura 3.20: Ejercicios rutina descargada



A pop-up form for creating a routine. It features a dark blue header with a white icon of a square with an arrow pointing right, and the text "Añadir rutina". Below the header is a light gray section with the label "Nombre" and a blue rectangular input field. At the bottom of this section is an orange button with the text "Añadir ejercicios".

Figura 3.21: Pop up crear rutinas

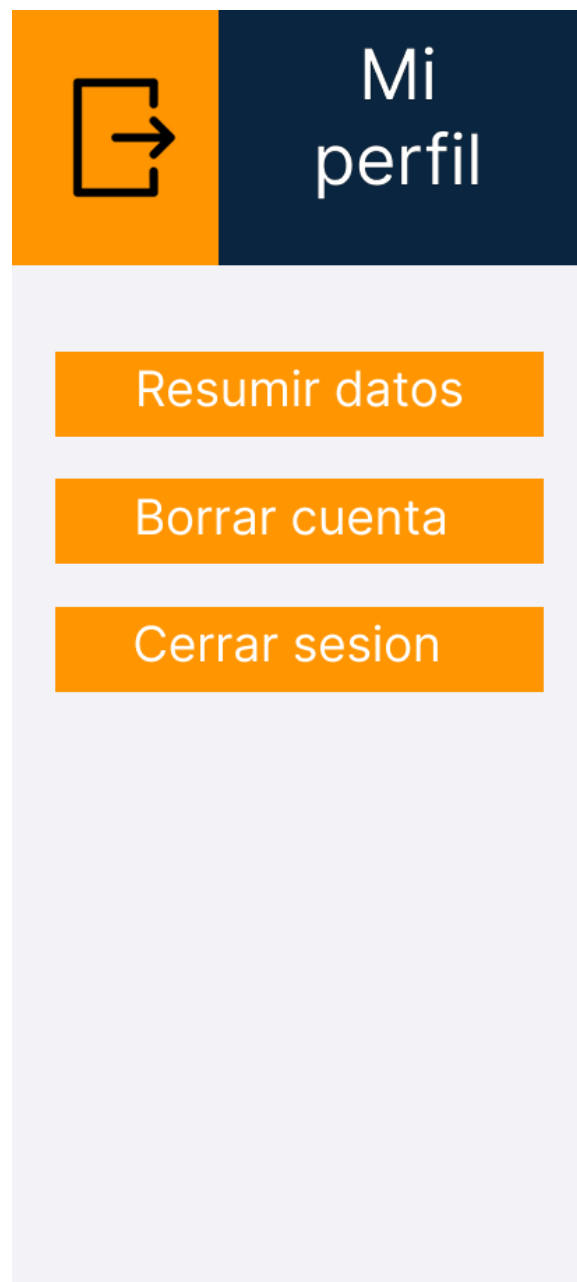


Figura 3.22: Opciones de perfil usuario



Figura 3.23: Pop up de confirmacion

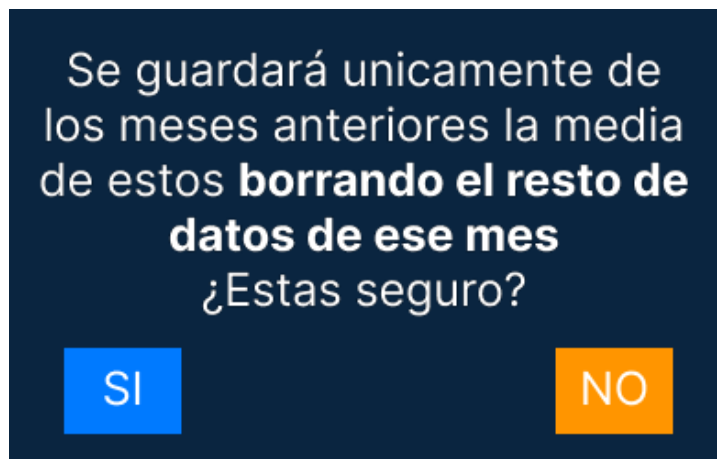


Figura 3.24: Pop up de confirmacion resumir datos

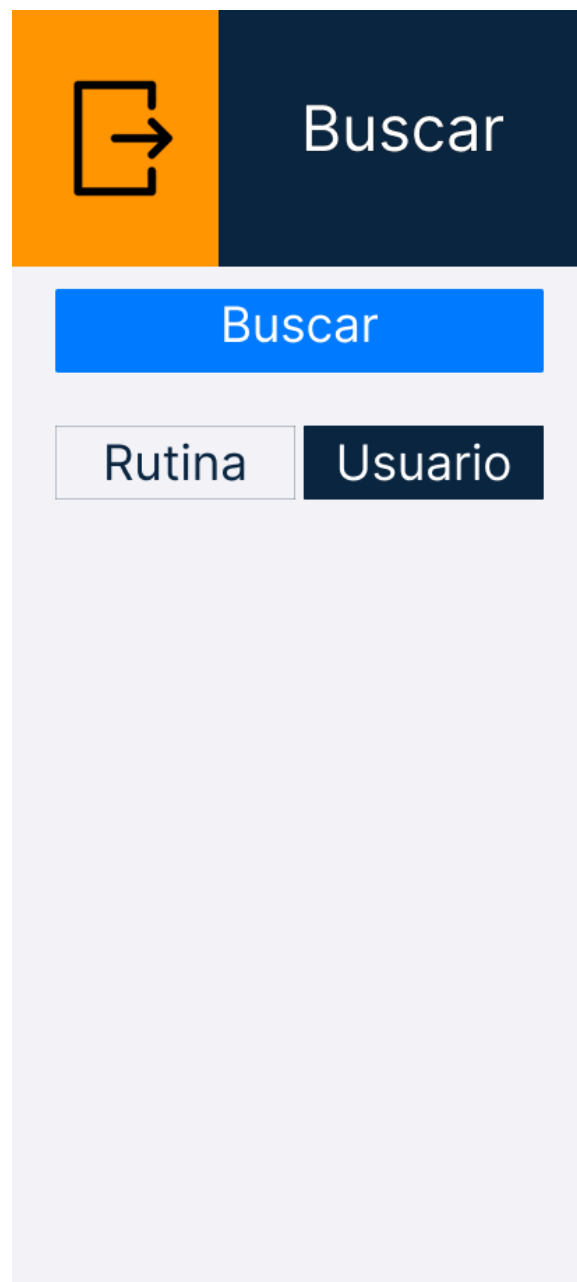


Figura 3.25: Buscar usuario



Figura 3.26: Buscar usuario filtrado

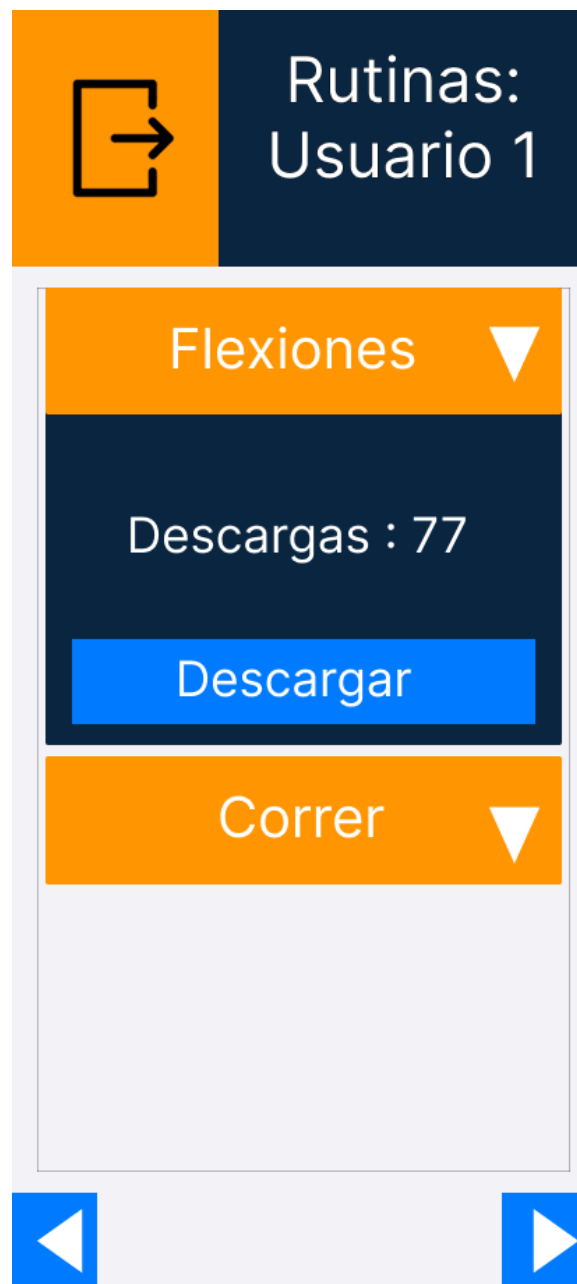


Figura 3.27: Rutinas de un usuario



Figura 3.28: Widget descargar rutina

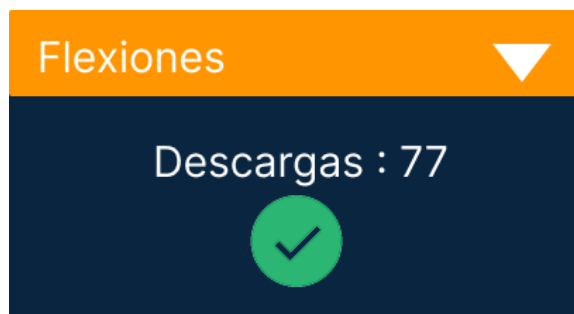


Figura 3.29: Widget rutina descargada





Figura 3.30: Buscar rutinas por su nombre filtrada

**Entrenamiento** (Una vez seleccionada una fecha en el calendario del menu principal saldría la siguiente ventana)

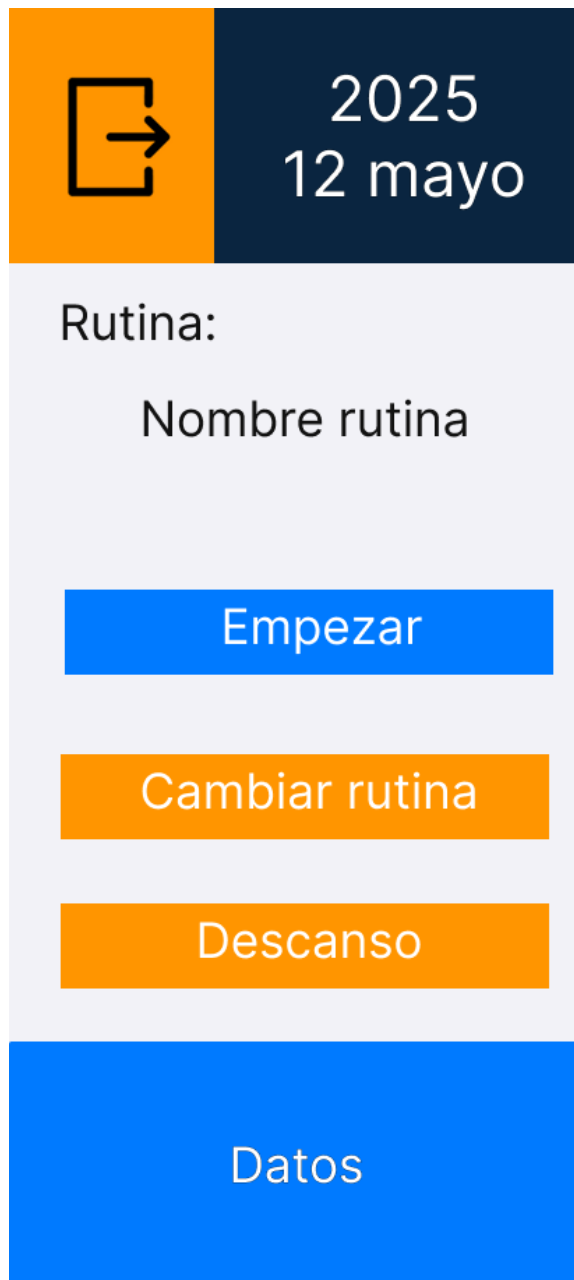


Figura 3.31: Entrenamiento de un día determinado

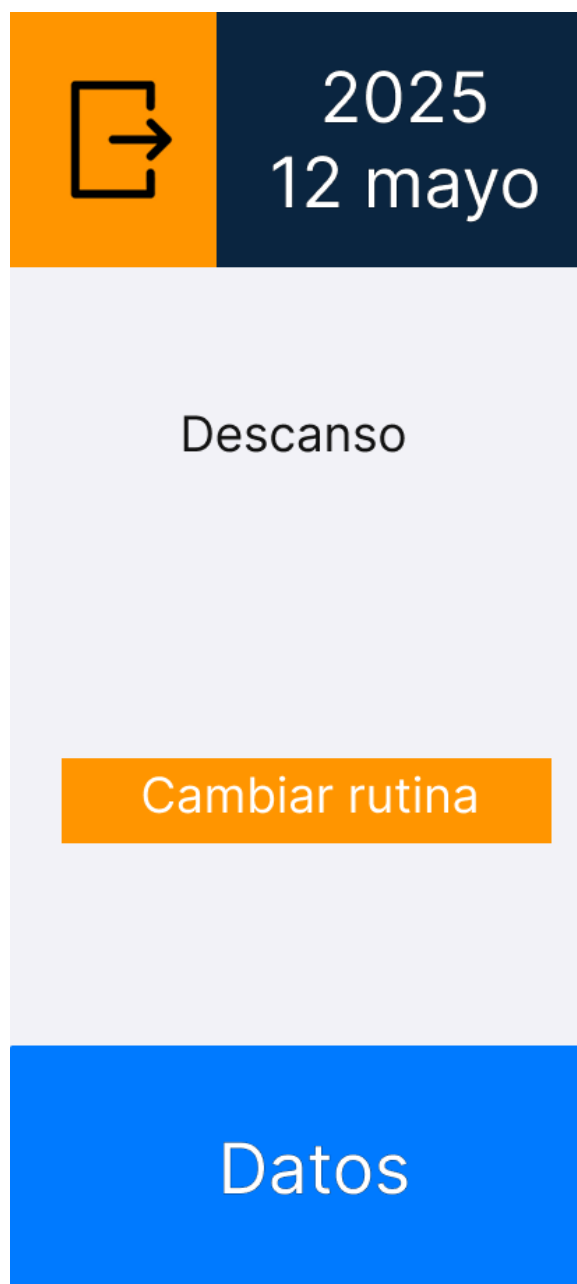


Figura 3.32: Descanso en un día determinado

Si seleccionamos datos de una fecha saldría esto

The image shows a mobile application interface with a header bar. The header bar is split into two sections: an orange section on the left containing a white icon of a square with an arrow pointing right, and a dark blue section on the right containing the text "2025" and "12 mayo" in white. Below the header, the main content area has a light gray background. It contains two input sections. The first section is titled "Peso corp: sin ingresar" in black text. Below this title is an orange button with the word "Ingresar" in white text. The second section is titled "Composicion corporal:" in black text, followed by the text "Músculo: X% Grasa: X% Osea: X%". Below this text is another orange button with the word "Ingresar" in white text.

Figura 3.33: Frame 29.png

La siguiente pantalla sale al comenzar el entrenamiento

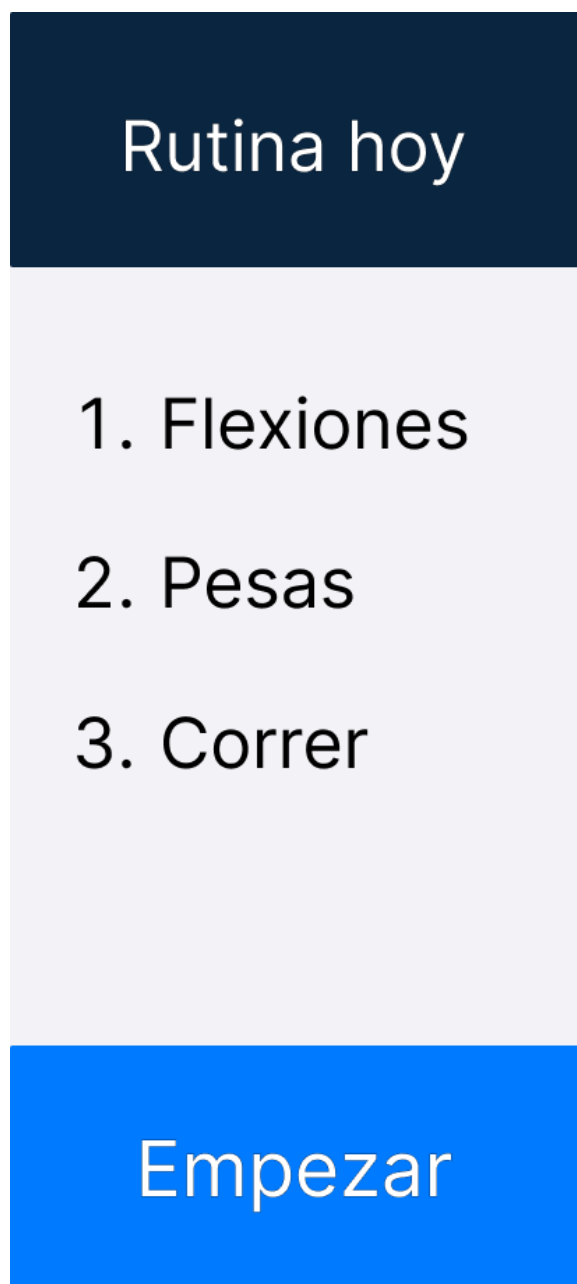


Figura 3.34: Lista ejercicios del entrenamiento actual

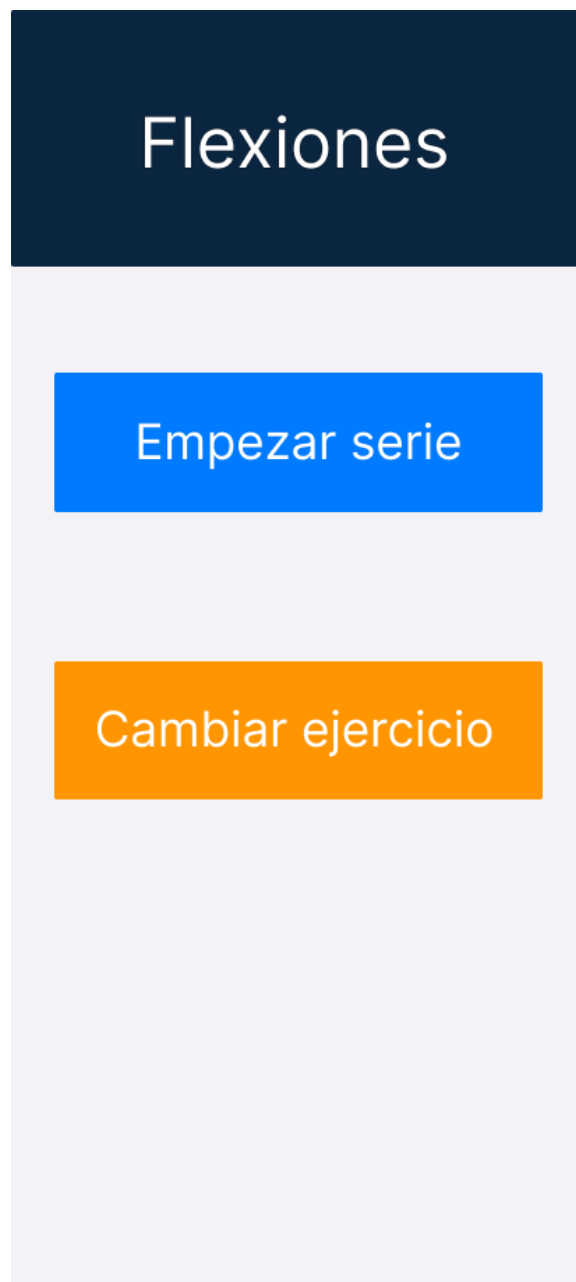


Figura 3.35: Primera serie flexiones

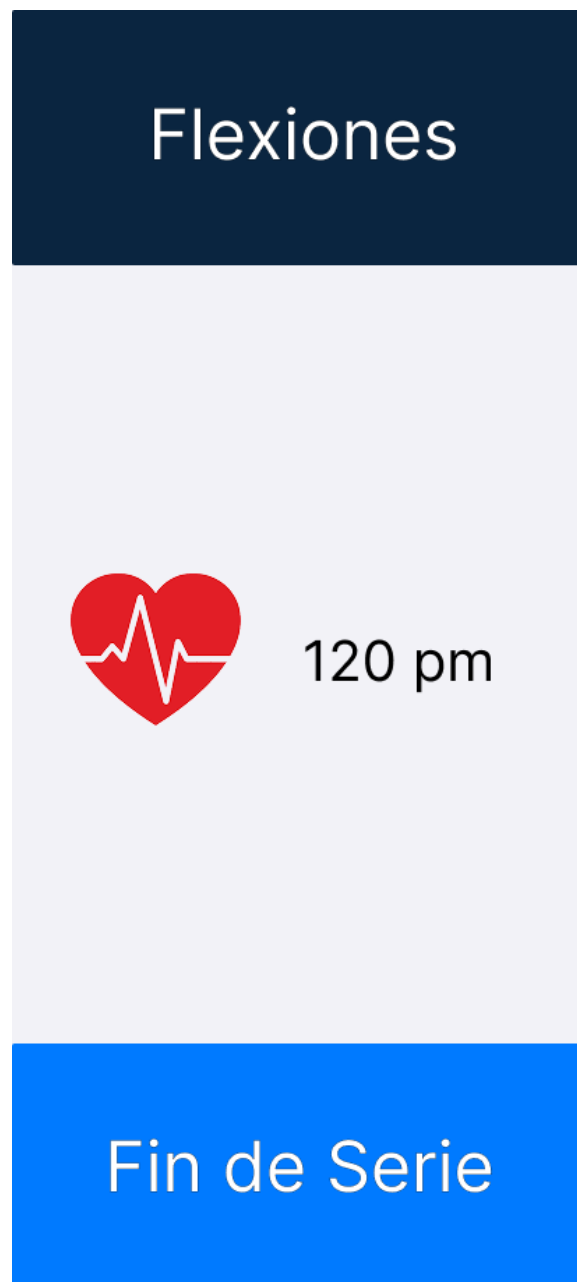


Figura 3.36: Realizando flexiones

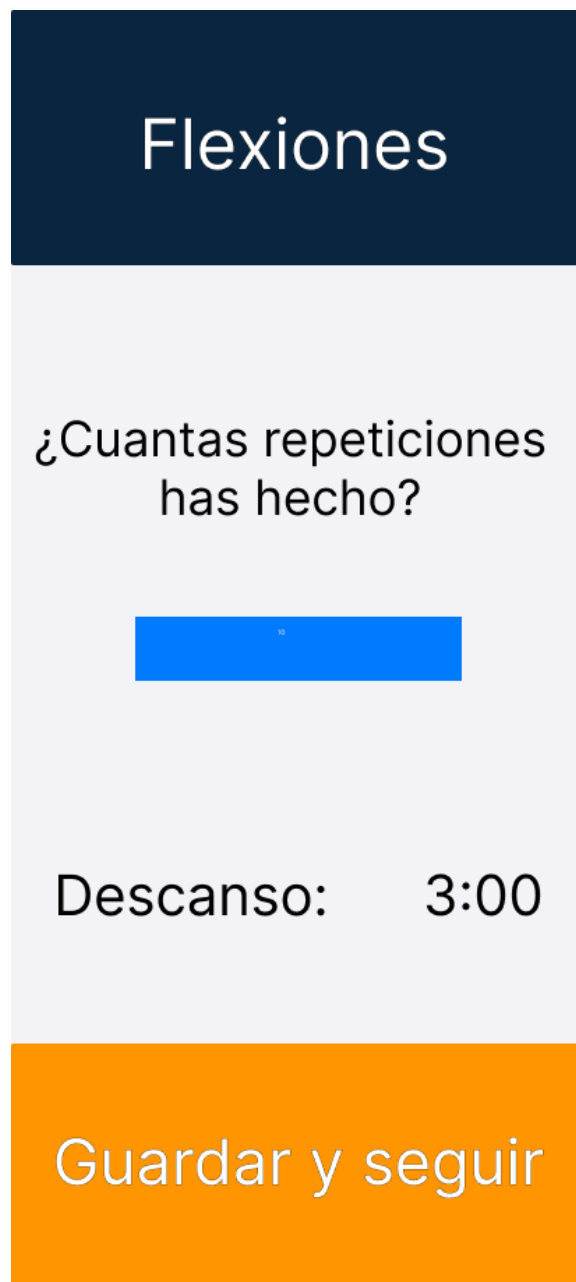


Figura 3.37: Fin de la serie(Descanso no completado)



The image shows a mobile application interface for a push-up series. It is divided into four horizontal sections. The top section is dark blue with the word 'Flexiones' in white. The second section is light gray and contains the question '¿Cuántas repeticiones has hecho?' followed by a blue input field with the number '10'. The third section is also light gray and shows 'Descanso: 0:00'. The bottom section is blue with the text 'Guardar y seguir' in white.

Flexiones

¿Cuántas repeticiones  
has hecho?

10

Descanso: 0:00

Guardar y seguir

Figura 3.38: Fin de la serie(Descanso completado)

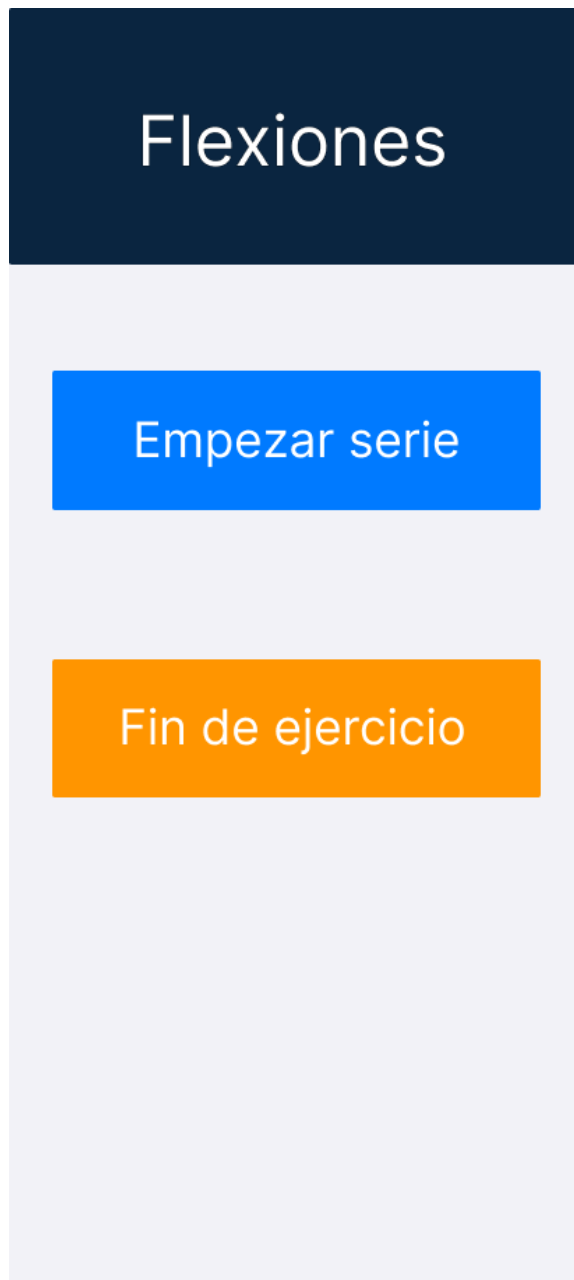


Figura 3.39: Acabar ejercicio o añadir serie

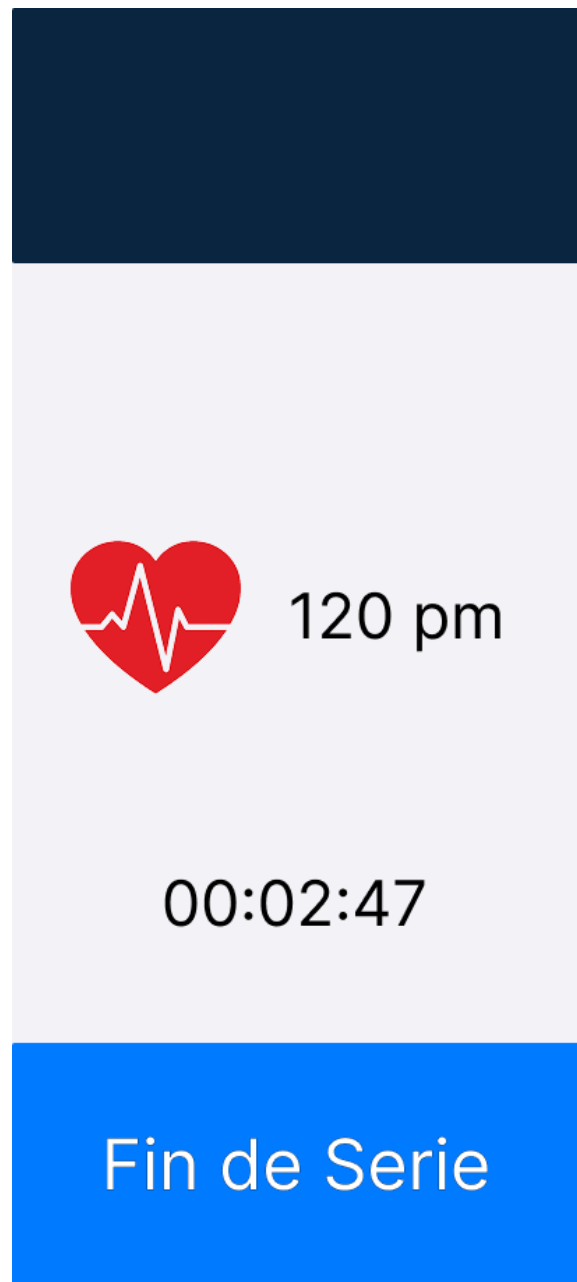


Figura 3.40: Realizando ejercicio midiendo tiempo

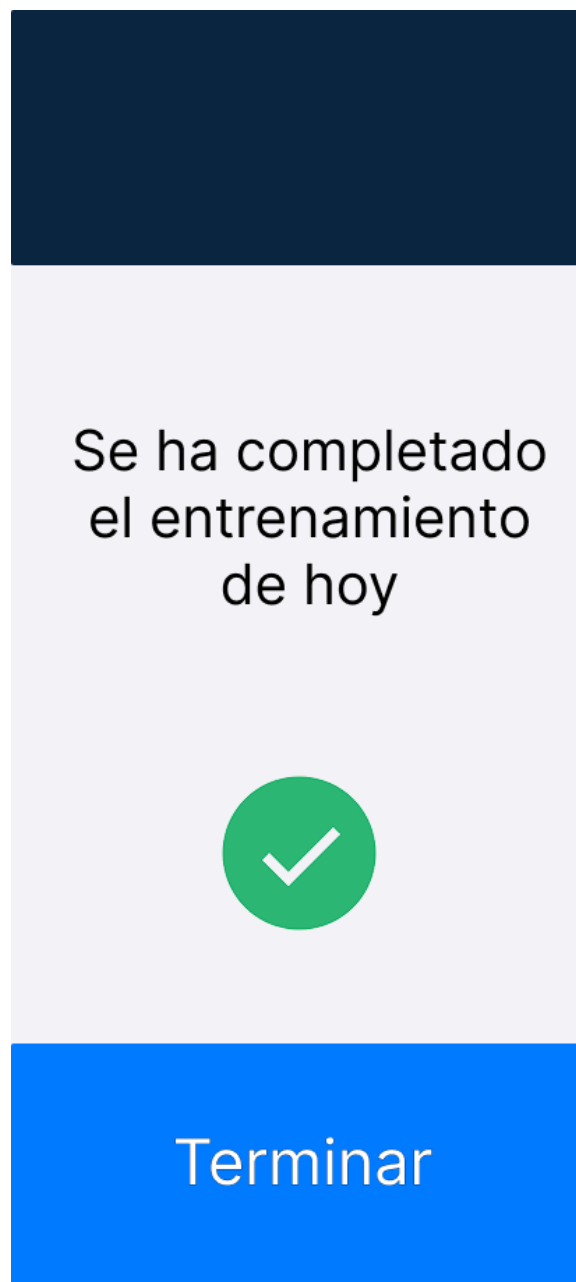


Figura 3.41: Fin entrenamiento

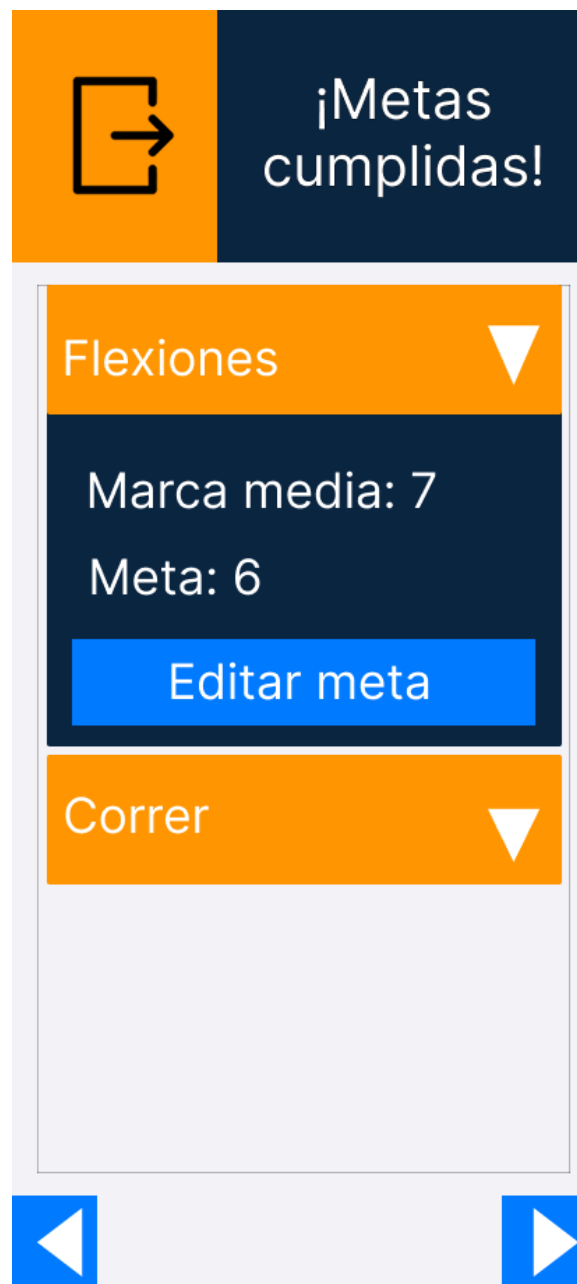


Figura 3.42: Metas cumplidas

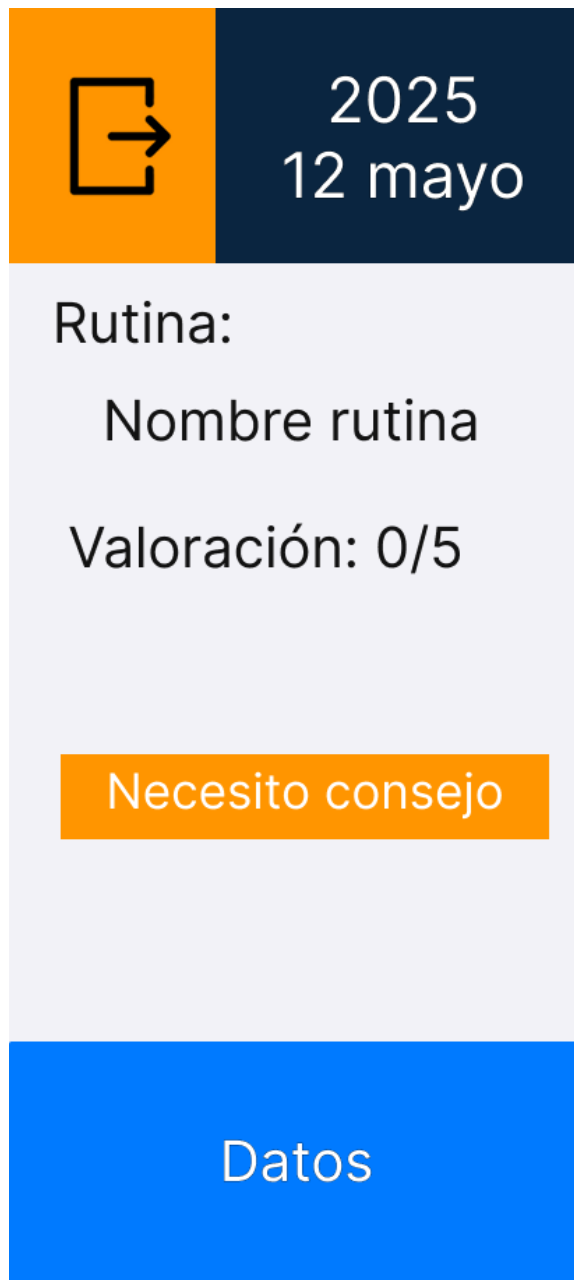


Figura 3.43: Datos entrenamiento terminado

El chat de la IA se abre cuando se le da a la opción necesito consejo de un entrenamiento realizado

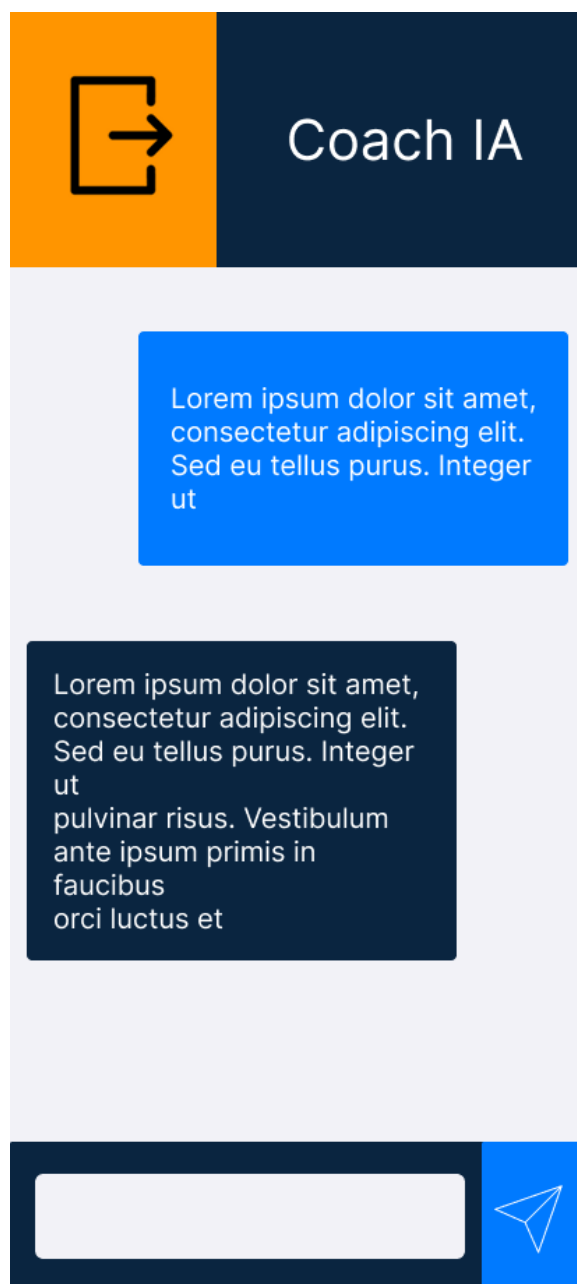


Figura 3.44: Chat con IA

Si le das a datos de un entrenamiento terminado

The image shows a mobile application interface for recording training data. At the top, there is a header bar with an orange square on the left containing a white icon of a document with an arrow pointing right, and a dark blue square on the right containing the text "2025" and "12 mayo" in white. Below the header, there is a light gray background area containing two orange rectangular buttons labeled "Ejercicio 1" and "Ejercicio 2" in white text. Below these buttons is a horizontal bar with a light gray center and blue squares on either side containing white left and right arrow icons. At the bottom, there is a dark blue rectangular area. Inside this area, the text "Peso corp: sin ingresar" is displayed in white. Below this text is an orange rectangular button labeled "Ingresar" in white text. Further down, the text "Composicion corporal:" is displayed in white, followed by "Músculo: X% Grasa: X% Osea: X%" in a smaller white font. At the bottom of this dark blue area is another orange rectangular button labeled "Ingresar" in white text.

Figura 3.45: Datos detallados entrenamiento terminado



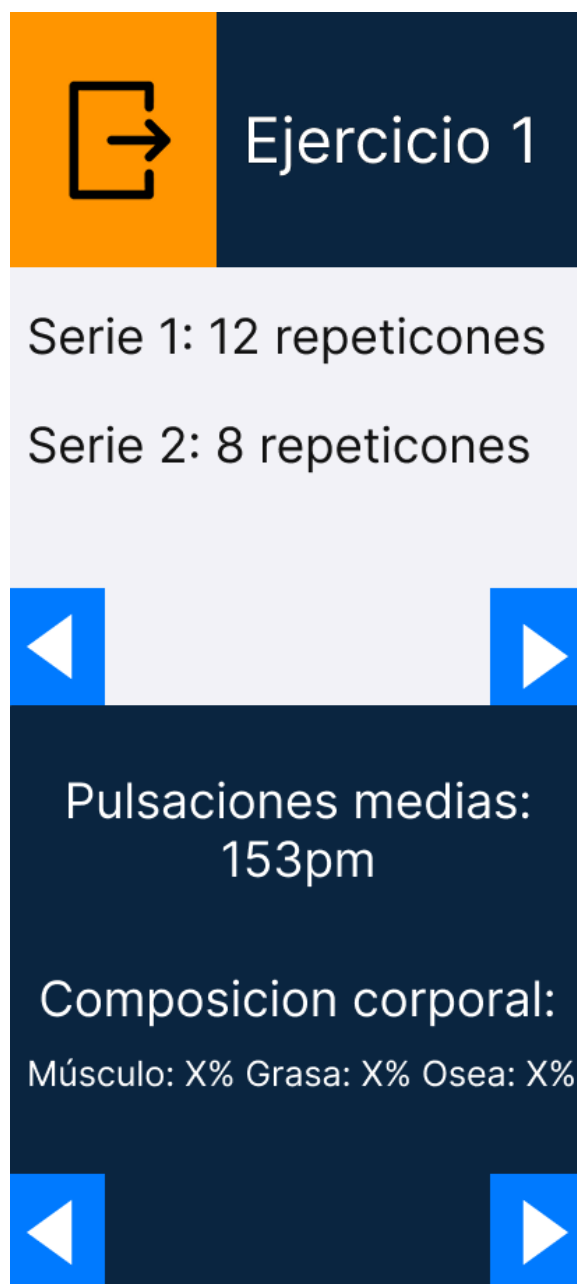


Figura 3.46: Pop up datos de un ejercicio en entrenamiento

Estos diseños pueden cambiar y sufrir alteraciones debido a los sprints reviews y correcciones de accesibilidad. De todas formas, cada cambio será informado con su motivo.

### 3.9.1.1. Sprint Review 0

Durante esta revisión de sprint se realizaron mejoras de diseño, como la incorporación de iconos en todos los botones para mejorar la accesibilidad. Se revisaron también los títulos de las ventanas, cambiando aquellos que no eran lo suficientemente claros.

Además, se añadieron nuevas historias al *product backlog*, incorporando o desglosando funcionalidades:

- SCRUM-42: Copiar rutina
- SCRUM-43: Añadir meta por parámetro
- SCRUM-44: Solicitar permiso al usuario antes de enviar datos a la IA
- SCRUM-45: Enviar datos del entrenamiento actual y anteriores a la IA
- SCRUM-46: Conectar/Desconectar con la IA

### 3.9.2. Iteración 1

La lista de historias de la primera iteración queda de la siguiente manera:

- SCRUM-26: Buscar ejercicio en lista
- SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
- SCRUM-27: Hacer la IU de la lista de ejercicios ELIMINAR
- SCRUM-28: Hacer la IU de creación de ejercicio ELIMINAR
- SCRUM-29: Pop up de confirmación
- SCRUM-30: Hacer IU de datos ejercicio ELIMINAR

A continuación especificaremos las tareas y pruebas de cada historia, y los resultados asociados a cada una.

SCRUM-26: Buscar ejercicio en lista
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la BD local</li> <li>2. Implementar algoritmo de búsqueda por nombre</li> <li>3. Implementar boceto de IU fig. 3.8</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si no existe ejercicio buscado, no sale nada en pantalla</li> <li>■ si existe ejercicio buscado, sale un acceso en pantalla</li> </ul>

<b>SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la BD local</li> <li>2. Implementar las funciones para las consultas</li> <li>3. Implementar las funciones para las modificaciones</li> <li>4. Implementar las funciones para el borrado</li> <li>5. Implementar boceto de la IU fig. 3.10, para borrar/modificar desde ahí</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se borra un ejercicio, no sale en la lista</li> <li>■ si se modifica, sale con los datos modificados</li> <li>■ si se inserta, sale en la lista el ejercicio</li> <li>■ se enseñan los datos del ejercicio de forma correcta</li> </ul>
<b>SCRUM-43: Añadir meta por parámetro</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la BD local</li> <li>2. Implementar las funciones para modificar metas</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si existe una meta determinada para ese ejercicio y se quiere insertar una, sustituir la actual</li> </ul>
<b>SCRUM-29: Pop up de confirmación</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la confirmación para que pueda ser usado en pantallas distintas</li> <li>2. Implementar boceto de IU fig. 3.23</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se ha seleccionado una opción, se devuelve la opción seleccionada</li> </ul>

Dentro del SCRUM-3 tenemos subfuncionalidades, funciones pequeñas que componen la grande:

### 3.9.2.1. Estado de la BD local

Al finalizar esta iteración, el estado de la BD queda tal y como se muestra en la figura

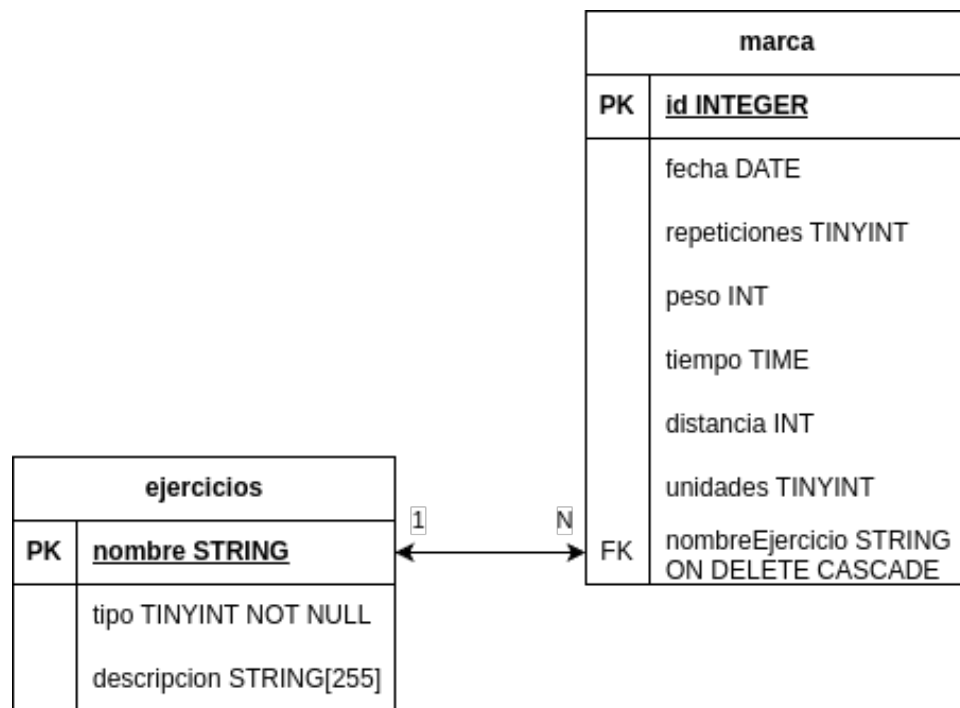


Figura 3.47: BD local iteracion 1

### 3.9.2.2. Sprint Review 1

Las primeras iteraciones tienden a ser más lentas debido a la fase inicial del desarrollo. No se completó la totalidad del sprint; quedó pendiente la subtask de modificar ejercicio en la base de datos (SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios). Aun así, las expectativas son positivas, ya que se espera un aumento en la velocidad de desarrollo.

La pantalla en la que se enseñan los datos de un ejercicio se modificó para facilitar su lectura y entendimiento, dado que su anterior diseño era confuso.

Otra parte a modificar es la del pop up de confirmación, ya que debido a la poca accesibilidad de los pop ups se sustituirá por una pantalla independiente.

### 3.9.3. Iteración 2

Esta iteración se centró en el desarrollo del sistema de sesiones, la API de la aplicación, el backend básico del servidor y la sección de rutinas. Las historias que incluye son las siguientes:

- SCRUM-21 Crear/Borrar mi usuario



Figura 3.48: Pantalla nueva

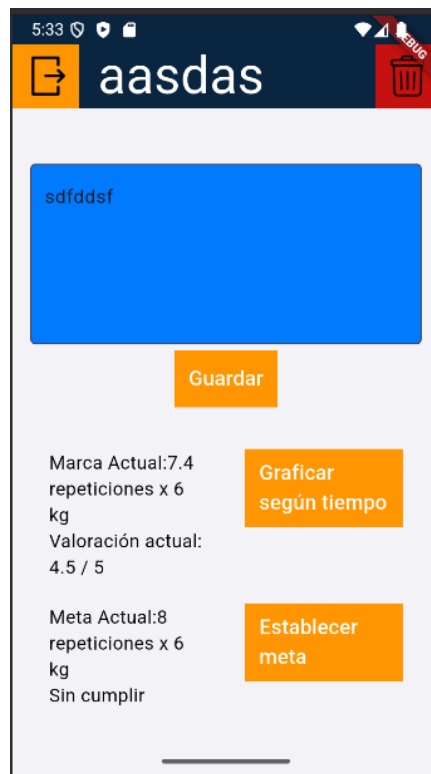


Figura 3.49: Pantalla antigua

Figura 3.50: Comparación entre la pantalla nueva y la anterior

- SCRUM-23 Iniciar/Cerrar sesión
- SCRUM-6 Insertar/Borrar/Modificar rutina
- SCRUM-4 Buscar rutina en la lista del usuario
- SCRUM-31 Crear IU de la lista de las rutinas ELIMINAR
- SCRUM-32 Crear IU de pantalla inicial ELIMINAR
- SCRUM-33 Implementar menú principal
- SCRUM-34 Crear IU datos rutinas ELIMINAR
- SCRUM-35 IU lista ejercicios de rutina modificable ELIMINAR
- SCRUM-36 Crear IU para crear rutina ELIMINAR

También se terminará la historia no finalizada de la iteración anterior(SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios).

SCRUM-33 Implementar menú principal
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar el boceto IU fig. 3.7</li> <li>2. <u>Añadir accesos a las funcionalidades actuales</u></li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si la funcionalidad seleccionada no está implementada, no hacer nada</li> <li>■ si la funcionalidad seleccionada está implementada, desplegar ventana de dicha funcionalidad</li> </ul>
SCRUM-21 Crear/Borrar mi usuario
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar bocetos de IU fig. 3.5 ,fig. 3.6, fig. 3.3 y la parte necesaria de fig. 3.22</li> <li>2. Implementar funciones de inserccion y borrado en el backend</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si el usuario que se quiere crear ya existe, informar y pedir que se rellenen los credenciales otra vez</li> <li>■ si el usuario que se quiere crear no existe, crear usuario y pasar a la siguiente pantalla</li> <li>■ si cualquier dato requerido está en un formato incorrecto ,informar al usuario para que vuelva a rellenar estos</li> <li>■ si todos los datos están en el formato correcto ,seguir con la creación del usuario</li> <li>■ si se ha creado exitosamente el usuario, volver a la pantalla inicial</li> <li>■ si se borra el usuario, borrar el token en el dispositivo y sus credenciales en el backend</li> </ul>
SCRUM-23 Iniciar/Cerrar sesión
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar el boceto IU fig. 3.4 y parte necesaria de fig. 3.22</li> <li>2. Implementar función de consulta en el backend</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si el usuario con esa contraseña no existe, informar al usuario</li> <li>■ si el usuario con esa contraseña existe, devolver un token para su guardado y guardar usuario</li> <li>■ si el usuario quiere cerrar sesión, borrar token y nombre del usuario del dispositivo y volver a la pantalla inicial fig. 3.3</li> </ul>

SCRUM-4 Buscar rutina en la lista del usuario
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar IU del boceto fig. 3.14</li> <li>2. Implementar lo necesario en la BD local</li> <li>3. Implementar algoritmo de búsqueda</li> <li>4. Implementar acceso a la rutina seleccionada</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si no existe la rutina buscada, no enseñar acceso en pantalla</li> <li>■ si existe la rutina buscada, enseñar acceso en pantalla</li> </ul>
SCRUM-6 Insertar/Borrar/Modificar rutina
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar las insercciones en para la BD local</li> <li>2. Implementar los borrados para la BD local</li> <li>3. Implementar las modificaciones para la BD local</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se hace una insercción, que el elemento insertado sea visible en la lista de ejercicios</li> <li>■ si se hace un borrado, que el elemento desaparezca de la lista</li> <li>■ si se hace una modificación, que se hagan visibles los cambios al momento</li> <li>■ si hay algún fallo en alguna de estas operaciones, informar al usuario con el motivo</li> </ul>

### 3.9.3.1. Sistema de sesiones

Dado a la complejidad del sistema de sesiones se le dedica este apartado de forma excepcional. Se implementó un sistema de autenticación basado en tokens. Cuando el usuario verifica su identidad, recibe un token que se guarda en el dispositivo y se usa para validar el acceso posterior. Esto evita que usuarios no autorizados suban contenido haciéndose pasar por otros.

El backend está desarrollado en *Node.js* y utiliza una base de datos SQL para gestionar las contraseñas de los usuarios. Este mismo backend se usará para las funcionalidades de las siguientes iteraciones.

### 3.9.3.2. API de la app

La comunicación entre la app y el servidor se realiza mediante peticiones HTTP. En el futuro, estas podrán migrarse a HTTPS para proteger operaciones

sensibles. Por ahora se mantiene HTTP para facilitar el depurado.

#### 3.9.3.3. Cambios importantes

Se rediseñó la implementación de las pantallas con listas. En lugar de una única clase con condiciones para la visibilidad de elementos, se optó por crear clases específicas para cada tipo de lista.

**Ventaja:** El código es más limpio, escalable y modular. Si hay que modificar un tipo de lista, los demás no se ven afectados.

#### 3.9.3.4. Sprint Review 2

Se sugirieron mejoras menores como el ajuste de tamaños e iconos en los botones de la sección de creación de rutinas.

También se modificó la funcionalidad de compartir rutinas. Ahora todas son modificables, eliminando la distinción entre rutinas descargadas (antes no modificables) y creadas (modificables). Esto mejora la experiencia del usuario: si desea volver a una rutina original, simplemente la puede volver a buscar y descargar.

Se cumplieron todas las funcionalidades en este sprint.



## 3.9.3.5. Estado de la BD local

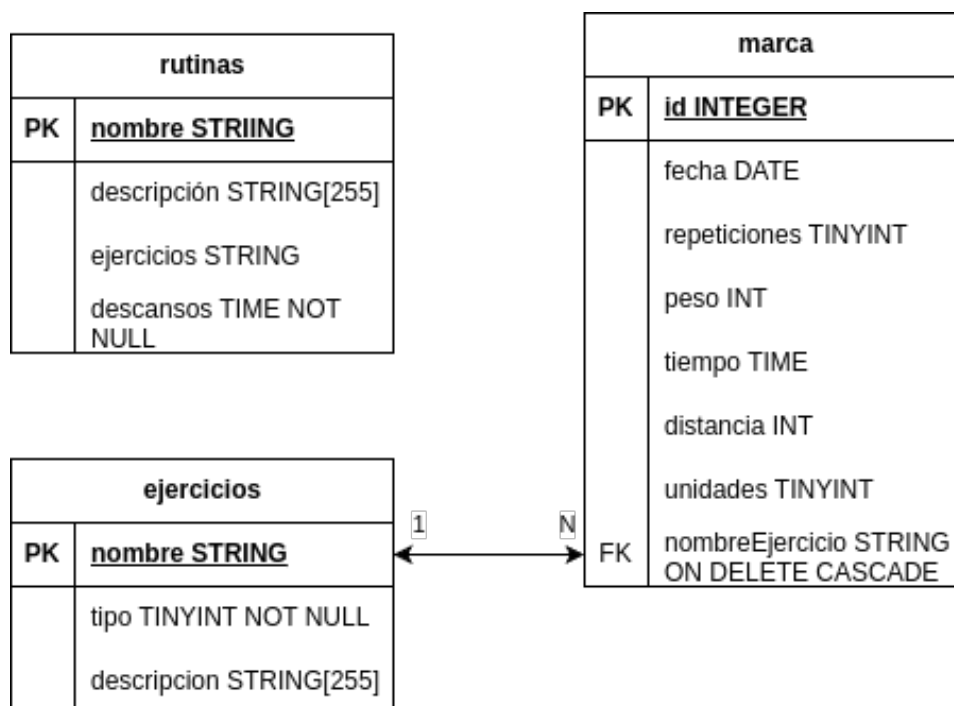


Figura 3.51: BD local iteracion 2

Se añadieron las tablas de ejercicios y marcas, una o varias marcas siempre están relacionadas con un ejercicio, de esta forma, se saben todas las marcas de un ejercicio y si se elimina un ejercicio sus marcas también desaparecen.

## 3.9.3.6. Estado de la BD backend

Antes de seguir, la BD del backend es donde se guardan los usuarios registrados y sus rutinas compartidas. Para diferenciarlo de la BD local, que es el almacenamiento interno del dispositivo.

usuario	
PK	<u>username VARCHAR[50]</u>
	passwd VARCHAR[255]

Figura 3.52: BD backend iteracion 2

#### 3.9.4. Iteración 3

Esta iteración se centró en desarrollar la funcionalidad para compartir rutinas entre usuarios, incluyendo la subida, visualización y descarga de rutinas. Además, se comenzó a implementar la funcionalidad de resumen de datos para optimizar el uso de la memoria local. Incluye las siguientes historias:

- SCRUM-2 Establecer peso objetivo
- SCRUM-37 Crear IU del perfil del usuario ELIMINAR
- SCRUM-42 Copiar rutina
- SCRUM-10 Enseñar datos de una rutina a descargar
- SCRUM-11 Compartir mi rutina
- SCRUM-39 Buscar rutinas para descargar
- SCRUM-9 Revisar datos para ver si el descanso es necesario
- SCRUM-22 Resumir datos
- SCRUM-40 IU del pop up de resumir datos ELIMINAR
- SCRUM-41 IU desplegable rutina ELIMINAR

<b>SCRUM-2 Establecer peso objetivo</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la parte necesaria de fig. 3.22</li> <li>2. Implementar el almacenaje de ese peso objetivo</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si el peso objetivo está en el formato correcto, guardar</li> <li>■ si el peso objetivo no está en el formato correcto, enseñar una alerta para su corrección</li> </ul>
<b>SCRUM-11 Compartir mi rutina</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la BD en el backend</li> <li>2. Implementar funciones de inserción en la API</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si la rutina no contiene ejercicios, no permitir su subida al servidor</li> <li>■ si la rutina contiene ejercicios, permitir la subida al servidor</li> </ul>
<b>SCRUM-39 Buscar rutinas para descargar</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar funciones de consultas en la API</li> <li>2. Implementar IU de búsqueda de rutinas por usuario creador y por nombre de rutina</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si busco un usuario que no existe, no se muestra ningún acceso en la IU</li> <li>■ si busco un usuario existente, se muestra su acceso en la IU</li> <li>■ si busco una rutina que no existe, no se muestra ningún acceso en la IU</li> <li>■ si busco una rutina existente, se muestra su acceso en la IU</li> </ul>
<b>SCRUM-10 Enseñar datos de una rutina a descargar</b>
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar el IU pop up</li> <li>2. Implementar las funciones de consulta en la API</li> <li>3. Implementar las funciones de inserción en la BD local</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se va a descargar la rutina y el usuario tiene en el dispositivo otra con el mismo nombre, un algoritmo resolverá este problema</li> <li>■ si se va a descargar la rutina y el usuario no tiene en el dispositivo otra con el mismo nombre, descargar normalmente</li> </ul>

SCRUM-42 Copiar rutina
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Añadir al boceto IU de datos rutina fig. 3.16 esta funcionalidad</li> <li>2. Implementar la IU resultante de la tarea anterior</li> <li>3. Implementar las consultas e inserciones necesarias en la BD local</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si hay algún problema por coincidencia de nombres, un algoritmo resolverá este problema</li> <li>■ si no hay algún problema por coincidencia de nombres, copiar</li> </ul>

#### 3.9.4.1. Funcionalidad descartada

Durante la realización de estas tareas de usuario, se dió notoriedad a que las historias restantes de este sprint (SCRUM-9 Revisar datos para ver si el descanso es necesario y SCRUM-22 Resumir datos) no pueden ser implementadas en este punto del desarrollo. Se pospondrán para su futura implementación.

Se ha decidido dar menos prioridad y que por tanto no se abordará en este TFG por limitaciones de tiempo en concreto el SCRUM-9 (Revisar datos para ver si el descanso es necesario), porque necesitaría los datos de un smartwatch y ha sido una funcionalidad previamente descartada.

También se ha reducido la prioridad de SCRUM-42 (Copiar rutina), dado que aporta poco valor a la app.

Se decidió adelantar otras, pero debido a los problemas encontrados durante esta iteración no se pudieron realizar.

#### 3.9.4.2. Peso objetivo

El peso objetivo es una meta que define el usuario y se almacena en memoria. Esta meta se representará en la gráfica de evolución del peso del usuario, y se generará una alerta cuando dicha meta sea alcanzada. La visualización aún no ha sido implementada completamente.

#### 3.9.4.3. Cambios durante el desarrollo

Durante esta iteración surgieron cambios en la forma de visualizar y gestionar las rutinas compartidas y almacenadas:

- Ahora, al descargar una rutina, se añade el nombre del creador al nombre de la rutina, facilitando la distinción entre rutinas propias y descargadas.

- Cuando hay conflicto de nombres al crear una rutina, se genera un nombre alternativo del tipo Nombre(n), siendo n un número incremental.
- Se ha implementado un selector entre “Mis rutinas” y “Compartidas” para facilitar la visualización de las rutinas subidas a la nube por el usuario.
- Se reemplazó el filtro integrado en la ventana de búsqueda por un selector emergente que permite buscar rutinas por nombre o por nombre de usuario.
- La interfaz de descarga y visualización de rutinas ahora es un cuadro emergente (pop-up) en lugar de un desplegable.

#### 3.9.4.4. Imprevistos y problemas en el desarrollo

Se identificaron dos errores principales de planificación:

**Error 1:** Se planificó implementar la funcionalidad de resumir datos sin haber completado la obtención de datos.

**Solución:** Replanificar los sprints.

**Error 2:** Se subestimó la complejidad del sistema de rutinas compartidas, especialmente en la resolución de conflictos de nombres.

**Solución:** Aplicar nombres combinados (nombre + creador + copia) en memoria local e identificadores *autoincrementales* en la nube.

#### 3.9.4.5. Valor añadido a la app

Los imprevistos y problemas surgidos durante el desarrollo han permitido detectar *bugs*, errores de diseño y carencias funcionales que de otro modo podrían haber pasado desapercibidos. Gracias a ello, se han podido proponer nuevas funcionalidades y mejorar las ya existentes, lo cual contribuye significativamente a aumentar la calidad global de la aplicación.

Algunas de las funcionalidades propuestas como mejora son:

- Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Al seleccionar la opción de borrar usuario del dispositivo, eliminar también la base de datos local asociada.
- Crear una base de datos local independiente para cada nuevo usuario creado en un dispositivo.
- Posibilidad de editar el nombre de una rutina.

- Marcar los ejercicios eliminados con una *flag*, para evitar que se inicien rutinas que los contengan.
- Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Estas funcionalidades están pensadas para aportar mayor **seguridad, integridad y calidad** al producto final.

#### 3.9.4.6. Base de datos del backend

La base de datos utilizada en el backend está basada en *MySQL*. En ella se almacenan los usuarios junto con sus contraseñas, así como los ejercicios y las rutinas. Las rutinas están vinculadas al usuario que las creó, y los ejercicios se asocian a las rutinas a las que pertenecen. Si un usuario decide eliminar su cuenta de forma permanente, las rutinas y ejercicios relacionados se eliminarán en cascada.

Al descargar una rutina, los ejercicios asociados se guardan en la memoria local del usuario como si fueran de su propiedad. Es decir, el usuario puede modificarlos libremente sin restricciones.

#### 3.9.4.7. Estado BD local

No ha habido cambios respecto a la iteración anterior

#### 3.9.4.8. Estado BD backend

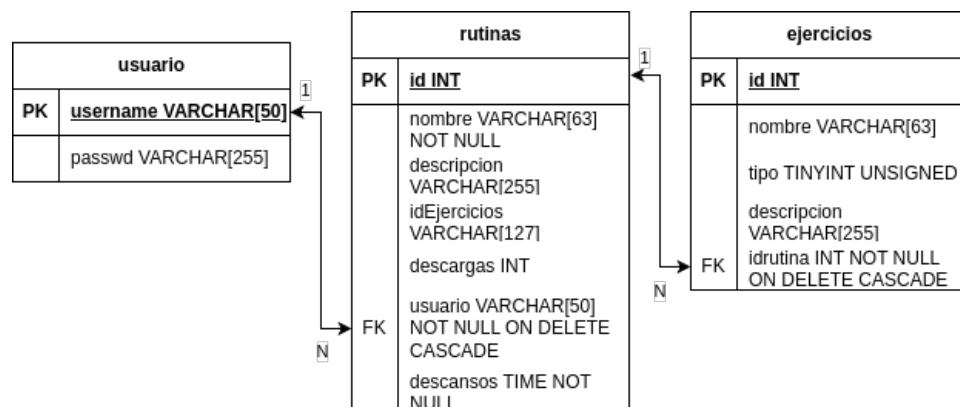


Figura 3.53: BD backend iteracion 3

#### 3.9.4.9. Sprint Review 3

En esta review se estuvo debatiendo si quitar las funcionalidades relacionadas con el smartwatch, por cuestiones de tiempo, ya que con las funcionalidades añadidas en el capítulo anterior se va ajustando el plazo.

En esta iteración se tenía pensado implementar la funcionalidad de resumir los datos, no obstante, al no tener implementada la parte en la que guardo las marcas de los entrenamientos, no puedo trabajar en esa funcionalidad. Por lo tanto, se ha pospuesto para la próxima iteración. También se le dió el visto bueno a la implementación de las funcionalidades previamente expuestas durante el desarrollo.

También, debido al querer hacer la app accesible se ha decidido sustituir los pop ups por pantallas normales, dado que los pop ups son un gran problema para este objetivo, debido a que un usuario puede hacerlo desaparecer este tocando fuera del area en pantalla de este.

#### 3.9.5. Iteración 4

En esta iteración se tiene pensado implementar la funcionalidad del calendario, donde el usuario podrá ver su planificación y acceder a sus marcas. Por otro lado, también se implementarán las funcionalidades nuevas del sprint anterior y las nuevas pantallas.

- SCRUM-39 Detectar metas cumplidas en los ejercicios despues del entrenamiento
- SCRUM-1 Registrar peso por día
- SCRUM-19 Realizar el flujo del entrenamiento
- SCRUM-44: Implementar calendario

SCRUM-44: Implementar calendario
<p>Tareas:</p> <ol style="list-style-type: none"><li>1. Implementar la IU e insertarla en el menu principal fig. 3.7</li><li>2. Implementar en la BD local</li><li>3. Implementar funciones de inserción en BD local</li><li>4. Implementar funciones de consulta en BD local</li><li>5. Implementar funciones de borrado en BD local</li><li>6. Implementar un acceso para ver los datos registrados sobre un día en concreto</li></ol>
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"><li>■ si un día no tiene una rutina asignada, se considera descanso</li><li>■ si un día tiene una rutina asignada, no hay marcas registradas y el día ya ha pasado, no se enseñará información, ya que se considera no entrenado</li><li>■ si un día tiene una rutina asignada, no hay marcas registradas pero es hoy, se enseñará un acceso para empezar el entrenamiento, se considera que el usuario todavía no ha entrenado</li><li>■ si un día tiene una rutina asignada, pero es futuro, no se enseñará un acceso para empezar el entrenamiento</li><li>■ si un día tiene una rutina asignada y hay marcas registradas sea el día actual o pasado, se mostrará información de las marcas, sin acceso a entrenamiento</li></ul>



<b>SCRUM-19 Realizar el flujo del entrenamiento</b>	
Tareas:	
<ol style="list-style-type: none"> <li>1. Implementar IU de las pantallas de entrenamientos(de la fig. 3.34 a la fig. 3.41)</li> <li>2. Implementar una clase que controle el cambio entre distintas pantallas y el guardado de marcas</li> </ol>	
Pruebas de aceptación:	
<ul style="list-style-type: none"> <li>■ si la rutina asignada para un día, no tiene ejercicios y se desea iniciar el entrenamiento, se muestra por pantalla una alerta y no se permite continuar con el entrenamiento</li> <li>■ si la rutina asignada para un día, tiene ejercicios, comienza el recorrido entre IUs</li> <li>■ si es la primera serie de un ejercicio, no se permite terminar el ejercicio</li> <li>■ si no es la primera serie de un ejercicio, se permite terminar el ejercicio</li> <li>■ si acaba una serie, no se permitirá acabar el ejercicio o continuar con este hasta que termine el tiempo de descanso que se muestra en pantalla</li> <li>■ si finaliza el último ejercicio de la lista, fin del entrenamiento, se devuelve al usuario al menú principal</li> </ul>	
<b>SCRUM-39 Detectar metas cumplidas en los ejercicios despues del entrenamiento</b>	
Tareas:	
<ol style="list-style-type: none"> <li>1. Implementar bocetos de IU fig. 3.42</li> <li>2. Implementar lógica para comparar marcas y metas despues de acabar el entrenamiento en la clase que controla el flujo del entrenamiento</li> </ol>	
Pruebas de aceptación:	
<ul style="list-style-type: none"> <li>■ si en un ejercicio no se ha cumplido la meta, no se enseña nada</li> <li>■ si en un ejercicio cumplimos una meta, se enseña en que ejercicio se cumplió la meta</li> </ul>	
<b>SCRUM-1 Registrar peso por día</b>	
Tareas:	
<ol style="list-style-type: none"> <li>1. Implementar boceto de IU fig. 3.45</li> <li>2. Implementar en BD local</li> <li>3. Implementar funciones de inserción</li> <li>4. Implementar funciones de consulta</li> <li>5. Implementar lógica para determinar si un usuario quiere subir o bajar de peso</li> </ol>	
Pruebas de aceptación:	
<ul style="list-style-type: none"> <li>■ si se introduce algún dato como formato incorrecto, se muestra una alerta por pantalla</li> <li>■ si se introducen los datos en formato correcto, se guarda</li> </ul>	

En esta iteración hay pocas funcionalidades comparadas con otras, porque el flujo del entrenamiento es una funcionalidad muy grande. A parte se añade esta funcionalidad que también se va a realizar en esta iteración, SCRUM-44 Implementar calendario.

### 3.9.5.1. Paquete tableCalendar

Es un paquete de flutter, que permite implementar y customizar un calendario, de forma fácil y rápida. También permite trabajar con información para reflejarla en las fechas del calendario.

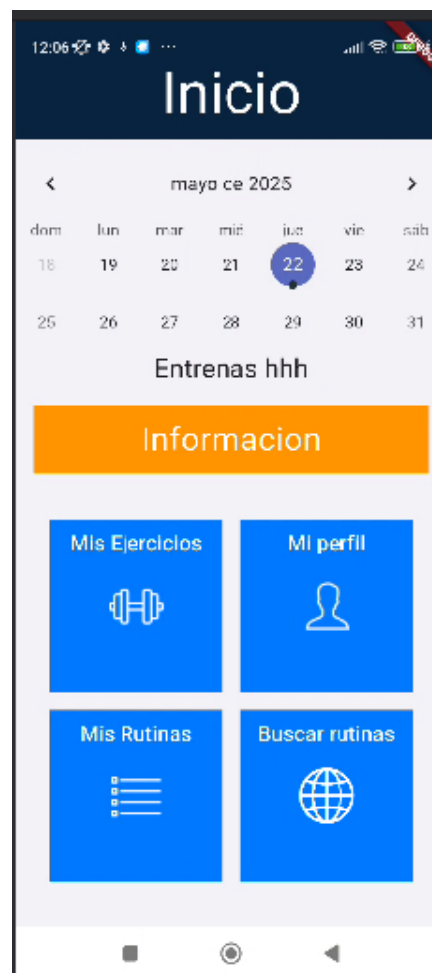


Figura 3.54: Pantalla Calendario

Existen varios formatos de calendario preimplementados, que cambian el nu-

mero de días en pantalla, se escogió este porque es el que menos problemas da en pantalla y más cómodo para las interacciones que busca el usuario.

#### 3.9.5.2. Pantallas cambiadas por pop ups

La solución que se ha dado para que este problema no quite mucho tiempo de desarrollo es la siguiente, reciclar todo el contenido posible del interior de los pop ups y plasmarlo tal cual en una pantalla. Dado que los diseños implementados eran buenos, quitando algún fallo que puedan dar por este cambio de formato.

Los pop ups que se cambiaron son los siguientes:

- Los pop ups de confirmacion
- Crear ejercicios
- Modificar ejercicios
- Nueva meta en un ejercicio
- Crear rutinas
- Modificar rutinas
- Modificar ejercicios de una rutina
- Información de una rutina a descargar
- Modificar informacion del perfil del usuario

Por consecuencia las futuras funcionalidades en la que se mencionan los pop ups, pasaran a ser pantallas completas.

#### 3.9.5.3. Cambios en el backend

Los cambios en el backend están relacionados con las siguientes funcionalidades propuestas en la iteracion anterior:

- Cambio 1: Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Cambio 2: Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Cambio 1: la solución implementada es la siguiente, a todas las funciones que puedan ser potencialmente víctimas de una suplantación(via petición normal de http), se les ha añadido un middleware, una función que se ejecuta antes para verificar el token. Express.js permite implementar esto rápidamente

Cambio 2: de primeras se pensó en usar una flag, pero finalmente para evitar pérdida de rendimiento al momento de que el usuario realice un entrenamiento(que es cuando en un principio se tenía pensado borrar el ejercicio), se optó por modificar la lista de ejercicios de la rutina al momento de borrar el ejercicio. El número de consultas iba a ser el mismo, pero se ahorra memoria, ya que no añadimos una columna más a la tabla de ejercicios

#### 3.9.5.4. Flujo entrenamiento

Para acaparar esta funcionalidad, se ha implementado una clase aparte encargada del desarrollo de esta parte de la app. Se ha decidido así ya que es una lógica más compleja y es conveniente tenerla separada de la IU.

Esta clase, solo se puede crear de forma útil llamando a un método estático de la propia clase (se hace así ya que necesita hacer operaciones asíncronas para obtener los datos necesarios), este método estático me devuelve la instancia con la que voy a trabajar. Se llama al método ejecutar de la instancia creada y sola se encarga de mostrar las pantallas al usuario, guardar los datos y de comparar las metas que se propuso el usuario.

Aquí un diagrama de actividades que explica la tarea de esta clase:

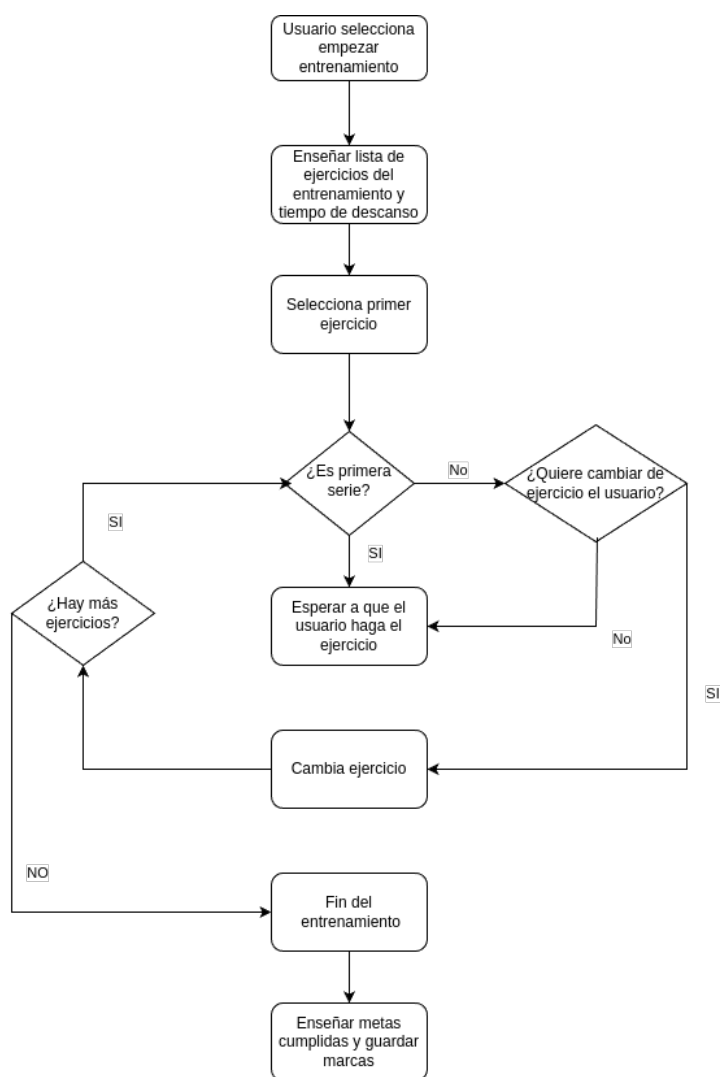


Figura 3.55: Flujo entrenamiento

## 3.9.5.5. Estado de la BD local

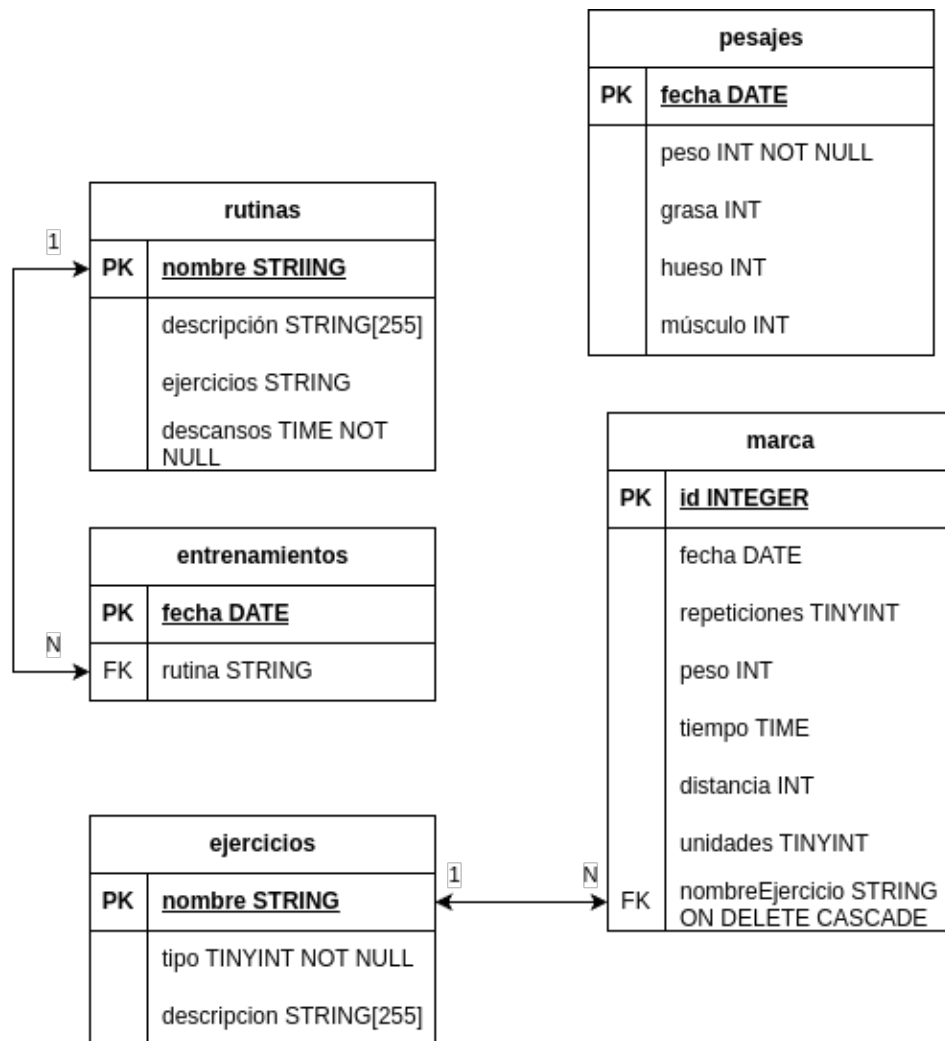


Figura 3.56: BD local iteración 4

## 3.9.5.6. Estado BD backend

No sufrió cambios respecto a la iteración anterior

#### **3.9.5.7. Sprint review 4**

En esta review se ha propuesto que después de los entrenamientos se les enseñe a los usuarios las metas que tenían establecidas siempre, aunque no la hayan superado, evitando siempre el feedback negativo al usuario.

#### **3.9.6. Iteración 5**

Debido a retrasos y problemas durante el desarrollo esta será la última iteración, que se dedicará a la implementación de gráficas para representar los datos requeridos por el usuario. Dado al tiempo limitado que se posee la implementación de la IA se aplazará al máximo e incluso siendo posible su descarte.